Efficient Semantic Segmentation for Real-time Applications

Von der Fakultät für Informatik
der Universität Duisburg-Essen

zur Erlangung des akademischen Grades

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Farnoush Zohourian
aus
Isfahan, Iran

1. Gutachter: Prof. Dr. Josef Pauli
2. Gutachter: Prof. Dr. Christian Wöhler

Tag der mündlichen Prüfung: 24/09/2024

DOI: 10.17185/duepublico/82745
URN: urn:nbn:de:hbz:465-20241218-071902-1

Acknowledgments

"The greatest glory in living lies not in never falling, but in rising every time we fall."(Nelson Mandela)

After almost spending seven years of so many rising ups and downs in my PhD journey, the time to reach the summit is closer than ever. The PhD process is incredibly daunting, challenging, intellectually stimulating, and emotionally draining experience. Therefore, to fulfil this mission requires the support of many people. There are ones, who could not have ended this difficult period without their support, guidance, help and love. First and foremost I am extremely grateful to my supervisors, Prof.Dr Josef Pauli for not only offering me the opportunity to do my Industrial-PhD in his department, but also for his invaluable advice, continuous support, guidance, motivation and patience, since I began my master study and passed several courses as well as doing my master thesis at his Intelligent Systems Chair. He always encouraged me in my whole academic life and I am really grateful of him for all collaboration during these years under his direction. I would like also to take this opportunity to thank Prof.Dr Christian Wöhler for agreeing to serve as external examiner for this academic work and participating in my doctoral examination.
I am indebted to APTIV (formerly Delphi) company and their managers and employees for serving as host and advisers to me for almost three years as I performed the core of my research as an external PhD student there. First, I would like to express my gratitude to Dr. Christian Nunn, the global manager of the computer vision and AI department in Aptiv, who gave me the opportunity to work in this high-tech company. In particular, I would like to thank Dr. Jan Siegemund and Dr. Borislav Antic, who graciously technically supported me and provided valuable feedback and guidance along my research endeavours in numerous ways. Their friendship, personal commitment and ongoing support always assisted me for the time I was there. My gratitude extends to all my managers, colleges and friends in APTIV for their kind help and support that have made my study and life in APTIV a cherished time spent together: Mirko Meuter, Dennis Müller, Patrik Gebhardt, Wolfgang Doerr, Lutz Roese-Koerner, Javier Marin, Yu Su, Honghui Jan, Stephanie Lessmann, Jens Westerhoff, Farzin Ghorban. I want to thank my parents for their love and support throughout my life. Thank you both for believed me in first place and allowing me to be as ambitious as I wanted and providing me a world full of opportunities. My siblings have been always my closest friends and biggest supporters. You, guys are awesome people and I am so glad to have you. The most special thank goes to my best partner and friend, my husband. Ali, your support, encouragement, unwavering love were undeniably what the God gifted me. Looking at you always pushed me forward. Thank you for unconditional love and help during this journey. Finally, I

also would like to thank my gorgeous daughter, Anita, who joined us in middle of my PhD, for giving me unlimited happiness and pleasure.

Farnoush Zohourian

Abstract

Scene understanding is a long-standing problem in the field of Computer Vision, which aims to extract useful information of a scene from raw sensor data and interpret the data content at the human understanding level. In the last decade, modern deep learning techniques enabled amazing developments of many Computer Vision tasks for purposive scene understanding at various levels of detail and abstraction. Semantic segmentation, as the main focus of this study, is an important tool for visual scene understanding that aims to assign a class label to each pixel in the image from a predefined set of classes. Semantic segmentation has shown critical usefulness to many applications, which need a precise, pixel-level understanding of their environment such as: autonomous vehicles, medical image analysis for computer-aided diagnosis, robot navigation, etc.
Recent advances in deep learning based semantic segmentation approaches show significant performance gains in terms of accuracy, which require high-end GPUs to run inferences in near real-time. However, some aspects of semantic segmentation such as computational efficiency have not been thoroughly studied. This is a challenging problem in many robotics platforms, where not only high-end GPUs are not always available, but also effectiveness and efficiency are highly required.
This thesis discusses our work towards solutions to the following challenges, namely 1) the time and computation constraints in real time applications or systems with limited computational power such as autonomous driving applications, 2) incorporating spatial relationship and contextual information, along with other high-level extracted features to improve scene understanding 3) model generalizability capability to work properly for unseen similar domain, when the labelled data available for training models is small or limited. Considering these problems by emphasising on solutions, which permit inference time reduction, the following contributions are presented in this thesis :

1. First, the thesis introduces a novel neural network-based semantic segmentation model, that is both memory-efficient and fast, capable of running on a CPU. This model is efficient both in execution time and memory requirements, which consumes very low computation resources and can be embedded in real-time systems. By integrating limited prior information, such as hand-crafted features, into the input data-model, it becomes possible to achieve excellent segmentation results without the necessity of increasing network layering. Moreover, this approach helps avoid the burden of a large number of parameters and reduces computational efforts. To showcase the practicality of this real-time CPU segmentation model, we apply it to the challenge of urban scene segmentation. Specifically, we focus on achieving efficient and highly accurate road segmentation, which holds significant potential for intelligent vehicle applications in

creating a safe drivable environment. It surpasses GPU-based state-of-the-art semantic segmentation performance at really fast rates.

2. To improve the precision of the proposed model, a graph-based image segmentation technique is employed, incorporating contextual information and spatial dependencies at minimal additional cost. Various optimization algorithms, including approximate inference procedures, are investigated to enhance the segmentation results within the specific graph-based neighbourhood model. The introduced method achieves state-of-the-art performance on benchmark datasets for road semantic segmentation, making it suitable for CPU or low-end GPUs.

3. The third contribution tackles three important challenges: a) the scarcity of training data, where only a small number of fully labelled images are available along with a large set of unlabelled data, b) the need to improve the generalizability of the model to work effectively on unseen but similar domains, and c) the necessity to enhance the model's robustness in the presence of context-changing factors, such as shadows on the road surface. To address these challenges, we propose a novel semi-supervised semantic segmentation method in conjunction with our previously introduced technique, resulting in fine-grained semantic segmentation results. This method leverages an unsupervised image-to-image translation technique, which aims to learn the translation between two visual domains without relying on paired data. By utilizing this approach, we demonstrate the effectiveness of our technique in road segmentation, a task known for its complexity due to the resemblance of roads to other patterns like walking areas, grass, and the presence of shadows or vehicles on the road surface. Notably, by addressing the issues of limited labelled data, enhancing generalizability, and improving robustness in the face of context-changing factors, our method achieves comparable performance to state-of-the-art methods, while operating efficiently on low-end GPUs, requiring low computational efforts.

Our methods have been tested on several public semantic segmentation datasets for autonomous driving and evaluated by well-known segmentation evaluation metrics. Experiments conducted on each method provide compelling evidence that all of our approaches produce more efficient semantic segmentation results compared to the state-of-the-arts methods.

Keywords: Semantic segmentation, Superpixels, Hand-crafted feature extraction, Graph-based image segmentation, Conditional Random Fields, Convolutional neural network, Semi-supervised semantic segmentation, Un-supervised image-to-image translation, Autonomous driving, Road segmentation

Kurzfassung

Szeneverständnis ist ein langjähriges Problem im Bereich der Computer Vision, das darauf abzielt, nützliche Informationen einer Szene aus rohen Sensordaten zu extrahieren und den Dateninhalt auf menschlicher Verständnisebene zu interpretieren. Im letzten Jahrzehnt haben moderne Deep-Learning-Techniken erstaunliche Fortschritte bei vielen Aufgaben der Computer Vision ermöglicht, um eine gezielte Szeneverständnis auf verschiedenen Detail- und Abstraktionsebenen zu erreichen. Die semantische Segmentierung, die im Mittelpunkt dieser Studie steht, ist ein wichtiges Werkzeug für das visuelle Szeneverständnis, das versucht, jedem Pixel im Bild eine Klassenbezeichnung aus einer vordefinierten Menge von Klassen zuzuweisen. Die semantische Segmentierung hat sich als äußerst nützlich für viele Anwendungen erwiesen, die ein präzises Verständnis auf Pixelebene ihrer Umgebung erfordern, wie z.B. autonome Fahrzeuge, medizinische Bildanalyse zur computerunterstützten Diagnose, Roboter-Navigation usw.
Aktuelle Fortschritte in Deep-Learning-basierten Ansätzen zur semantischen Segmentierung zeigen signifikante Leistungssteigerungen in Bezug auf Genauigkeit, erfordern jedoch leistungsstarke GPUs, um Inferenzen nahezu in Echtzeit auszuführen. Allerdings wurden einige Aspekte der semantischen Segmentierung, wie z.B. die Rechenleistung, nicht gründlich untersucht. Dies stellt ein herausforderndes Problem in vielen Robotik-Plattformen dar, in denen nicht nur leistungsstarke GPUs nicht immer verfügbar sind, sondern auch Effektivität und Effizienz stark gefordert werden.
Diese Arbeit diskutiert unsere Lösungsansätze für die folgenden Herausforderungen: 1) Zeit- und Rechenbeschränkungen in Echtzeitanwendungen oder Systemen mit begrenzter Rechenleistung, wie z.B. autonome Fahrzeuganwendungen, 2) Integration von räumlichen Beziehungen und kontextuellen Informationen zusammen mit anderen hochrangigen extrahierten Merkmalen, um das Szeneverständnis zu verbessern, 3) Fähigkeit des Modells zur Generalisierung, um ordnungsgemäß in unbekannten, ähnlichen Domänen zu funktionieren, wenn die für das Training verfügbaren gelabelten Daten klein oder begrenzt sind. Unter Berücksichtigung dieser Probleme, indem Lösungen betont werden, die eine Reduzierung der Inferenz-zeit ermöglichen, werden in dieser Arbeit folgende Beiträge vorgestellt:

1. Erstens wurde in der Arbeit ein neuartiges semantisches Segmentierungsmodell basierend auf neuronalen Netzwerken eingeführt, das sowohl speicher-effizient als auch schnell ist und auf einer CPU ausgeführt werden kann. Dieses Modell ist sowohl in Bezug auf die Ausführungszeit als auch die Speicheranforderungen effizient und verbraucht sehr geringe Rechenressourcen, wodurch es in Echtzeitsysteme integriert werden kann. Durch die Integration begrenzter vorheriger Informationen, wie z. B. manuell definierte Merkmale, in das Eingabedatenmodell können ausgezeichnete Segmentierungsergebnisse erzielt werden, ohne

die Notwendigkeit einer Erhöhung der Netzwerkschichtung. Darüber hinaus hilft dieser Ansatz, die Belastung durch eine große Anzahl von Parametern zu vermeiden und den Rechenaufwand zu reduzieren. Um die Praktikabilität dieses Echtzeit-CPU-Segmentierungsmodells zu demonstrieren, wird es auf das Problem der Segmentierung städtischer Szenen angewendet. Insbesondere konzentrieren wir uns darauf, eine effiziente und hochpräzise Straßensegmentierung zu erreichen, die ein erhebliches Potenzial für intelligente Fahrzeuganwendungen zur Schaffung einer sicheren befahrbaren Umgebung birgt. Das Modell übertrifft die leistungsstärkste semantische Segmentierung auf GPU-Basis mit wirklich schnellen Raten.

2. Zweitens wird zur Verbesserung der Präzision des vorgeschlagenen Modells eine graphbasierte Bildsegmentierungstechnik eingesetzt, die Kontextinformationen und räumliche Abhängigkeiten mit minimalen zusätzlichen Kosten berücksichtigt. Verschiedene Optimierungsalgorithmen, einschließlich approximativer Inferenzverfahren, werden untersucht, um die Segmentierungsergebnisse innerhalb des spezifischen graphbasierten Nachbarschaftsmodells zu verbessern. Die vorgestellte Methode erzielt eine Leistung auf dem Stand der Technik bei Benchmark-Datensätzen für die semantische Segmentierung von Straßen und eignet sich für CPUs oder Low-End-GPUs.

3. Der dritte Beitrag befasst sich mit drei wichtigen Herausforderungen: a) dem Mangel an Trainingsdaten, bei dem nur eine kleine Anzahl vollständig gekennzeichneter Bilder zusammen mit einer großen Menge an nicht gekennzeichneten Daten zur Verfügung steht, b) der Notwendigkeit, die Generalisierbarkeit des Modells zu verbessern, um effektiv in unsichtbaren, aber ähnlichen Bereichen zu arbeiten, und c) der Notwendigkeit, die Robustheit des Modells angesichts von Kontextänderungen, wie z. B. Schatten auf der Straßenoberfläche, zu verbessern. Um diesen Herausforderungen zu begegnen, schlagen wir eine neuartige halb-überwachte semantische Segmentierungsmethode in Verbindung mit unserer zuvor vorgestellten Technik vor, um feinkörnige semantische Segmentierungsergebnisse zu erzielen. Diese Methode nutzt eine unbeaufsichtigte Bild-zu-Bild-Übersetzungstechnik, die darauf abzielt, die Übersetzung zwischen zwei visuellen Domänen zu erlernen, ohne auf gepaarte Daten angewiesen zu sein. Durch die Anwendung dieses Ansatzes demonstrieren wir die Wirksamkeit unserer Technik bei der Straßensegmentierung, einer Aufgabe, die aufgrund der Ähnlichkeit von Straßen zu anderen Mustern wie Gehbereichen, Gras und dem Vorhandensein von Schatten oder Fahrzeugen auf der Straßenoberfläche bekanntermaßen komplex ist. Beachtenswert ist, dass unsere Methode durch die Bewältigung der Probleme begrenzter gekennzeichneter Daten, Verbesserung der Generalisierbarkeit und Steigerung der Robustheit angesichts von kontextverändernden Faktoren eine vergleichbare Leistung wie modernste Methoden erzielt, während sie effizient auf Low-End-GPUs arbeitet und geringen Rechenaufwand erfordert.

Unsere Methoden wurden an mehreren öffentlichen semantischen Segmentierungsdatensätzen für autonomes Fahren getestet und anhand bekannter Segmentierungsbewertungsmetriken bewertet. Experimente, die mit jeder Methode durchgeführt wurden, liefern überzeugende Beweise dafür, dass alle unsere Ansätze effizientere semantische Segmentierungsergebnisse im Vergleich zu den Methoden nach dem Stand der Technik liefern.

Schlüsselwörter: Semantische Segmentierung, Superpixel, Handgefertigte Merkmalsextraktion, Graphbasierte Bildsegmentierung, Bedingte Zufallsfelder, Convolutional Neural Network, Semi-überwachte semantische Segmentierung, Unüberwachte Bild-zu-Bild-Übersetzung, Autonomes Fahren , Straßensegmentierung

My related Publications and Patents

Major parts of this thesis have been published in peer-reviewed journals, conference proceedings and patents. In this section, I provide a comprehensive list of my related publications and patent, highlighting their relevance to specific contributions or chapters of my thesis. This list reflects the importance of my research efforts and their alignment with the main themes of my work.

1 Publications

The following is a chronological list of my publications, categorizing them as journal articles and conference papers.

1.1 Journal Article

- Farnoush Zohourian and Josef Pauli, "Coarse-to-Fine Semantic Road Segmentation Using Super-Pixel Data Model and Semi-Supervised Modified CycleGAN," Journal of Image and Graphics, Vol. 10, No. 4, pp. 132-144, December 2022.
  - Related Contribution/Chapter: Chapter 5

1.2 Conference Papers

- Zohourian, Farnoush, Borislav Antic, Jan Siegemund, Mirko Meuter, and Josef Pauli. "Superpixel-based Road Segmentation for Real-time Systems using CNN." In VISIGRAPP (5: VISAPP), pp. 257-265. 2018.
  - Related Contribution/Chapter: Chapter 3
- Zohourian, Farnoush, Jan Siegemund, Mirko Meuter, and Josef Pauli. "Efficient fine-grained road segmentation using superpixel-based CNN and CRF models." In International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI 2018). CENPARMI, Centre for Pattern Recognition and Machine Intelligence Concordia University, Montreal, Canada, 2018, pp. 512–517.
  - Related Contribution/Chapter: Chapter 4
- Zohourian, Farnoush, and Josef Pauli. "Enhancement of the super pixel-CNN based road segmentation using cycle consistent adversarial network." In Fourteenth International Conference on Machine Vision (ICMV 2021), vol. 12084, pp. 91-100. SPIE, 2022.

  - Related Contribution/Chapter: Chapter 5
  - **Award:** This publication was honored with the **Best Session Presentation Award** at the ICMV 2021. Details of the award can be found at https://icmv.org/photo2021.html.

2 Patents

I have also made a significant contribution to the development of intellectual property through the following patent:

- Zohourian, Farnoush, Antic, B., Siegemund, J. and Meuter, M., Aptiv Technologies Ltd and Delphi Technologies LLC, 2018. Method for the semantic segmentation of an image. U.S. Patent Application 15/949,246.
  - Related Contribution/Chapter: Chapter 3

Table of Contents

1 Introduction

Visually perceiving natural scenes, where a human can act within or navigate, and obtaining a comprehensive understanding of high-level scene structures can quickly be performed by the human visual system. However, providing the similar capability to understand a 3D scene represented in a 2D image or video by machines and computer systems is the most challenging problem. Understanding a scene is much more than to simply record and store it, extract some features, and eventually recognize an object. It integrates meaningful information at multiple levels to find a mapping to extract semantic relationships and patterns from sensor data. The field of computer vision encompasses a wide range of problems and tasks aimed at enabling machines to achieve purposive scene understanding at varying levels of detail and abstraction by extracting information from visual data. To achieve a comprehensive understanding, it is necessary to strike a delicate balance between local, global, and dynamic aspects that may exist within a scene. For instance, an observer may be interested in determining the presence of a car in an image (recognition), localizing the exact position of the car (localization), or even counting the number of car instances in the image (instance segmentation). Another task could involve recognizing objects that have never been seen before by observing a set of similar objects. Developing computing machines capable of effortlessly performing these vision tasks poses significant challenges. Based on the level of complexity, semantic scene information can be roughly divided into different levels. These levels include Scene Classification, which provides a single tag or label for the entire image. There is also Object Detection within the scene, identifying bounding boxes for different objects in the image. Scene Semantic Segmentation is another level, where a per-pixel semantic tag is assigned to the image. Lastly, there is Instance Segmentation, which not only assigns a per-pixel semantic tag but also a unique object identifier to each distinct object of interest in the image.

This thesis deals with a fundamental task in computer vision, i.e. the semantic segmentation, which is widely regarded as the main step of image analysis and scene understanding and has shown favourable usefulness to many applications such as: medical image analysis for computer-aided diagnosis, robot navigation, video surveillance , human-computer interaction and virtual reality and self-driving vehicles. Figure. 1.1 shows semantic segmentation in some applications. In autonomous driving, as our case study, semantic segmentation provides the essential understanding of constantly-changing environment for making decisions, like the recognition of objects (buildings, road, lane markings), shape (cars, pedestrians) and the spatial relationship (differentiating between road and side-walk or grass). The goal is to assign a semantic label to each pixel corresponding to objects contained in an image. Our fo-

cus is on the utilization of advanced machine learning techniques to solve the problem in a cost-effective computational approach.

1.1 Background

Most traditional image segmentation methods attempt to group pixels to define "coherence" in terms of low-level cues such as intensity, texture, spatial information, and smoothness of boundary [Achanta et al., 2012, Comaniciu and Meer, 2002, Garcia-Lamont et al., 2018]. In these methods the obtained segmented image is mostly covered with regions that are not completely homogenous. A segmented region could cover more than one object, or one object could be divided into several regions.
Image segmentation using graphical models such as Markov Random Field (MRF) or the Conditional Random Field (CRF) [Camilus and Govindan, 2012, Liu et al., 2017, Kohli and Torr, 2007, Arnab et al., 2018, Zhou et al., 2016] is another type of image segmentation method, where each pixel is assigned to a specific class label in such a way, that pixels with certain object characteristics belong to the same class. The model defines the joint probability distribution of the image observation and the label random variables on the 2D regular lattice. Non-causal relationships among the nodes in a graphical model, such as the spatial relationships among neighbouring labels, are extracted to force the adjacent pixels to be classified into the same group and calculate the energy functions for the graphical models. Although, much research have been followed on the basic framework such as incorporating the hierarchical connectivity and higher-order potentials to improve the segmentation and labelling accuracy, however they are suffering from modelling complexity and high computational cost.
In recent years, deep learning techniques and in particular, Convolutional Neural Networks (CNNs) [LeCun et al., 1998, Krizhevsky et al., 2012, O. Russakovsky, 2015, LeCun et al., 2015, Szegedy et al., 2015, He et al., 2016] made a breakthrough in effectiveness of image analysis and semantic segmentation. They enable, through different network layering, the automatic learning of hierarchical image features from low-level to high-level. The early CNN-based semantic segmentation methods used a patch-wise classification technique, where a sliding window moves across the image and splits it as many as local overlapped patches. These studies generally required a refinement process, since labelling assigning was done separately for each patch and the segmentation results must be aggregated to achieve the same resolution as the input, which mostly produced undesirable segmentation results [Ulku and Akagündüz, 2022, Lateef and Ruichek, 2019].
Pixel-wise segmentation based on Fully Convolutional Neural Networks (FCNs) [Long et al., 2015] is another type of semantic segmentation method, which conserve the spatial information of their inputs throughout the computation i.e., their inputs and outputs are of the same spatial size. The network takes an image as input and outputs the probability maps of each pixel belonging to pre-defined object classes. The Encoder-Decoder based architecture, known as the U-nets [Ronneberger et al., 2015], is the state-of-the-art image segmentation study following the fully convolutional construction. Its network is comprised of two parts. Encoder part gradually reduces the

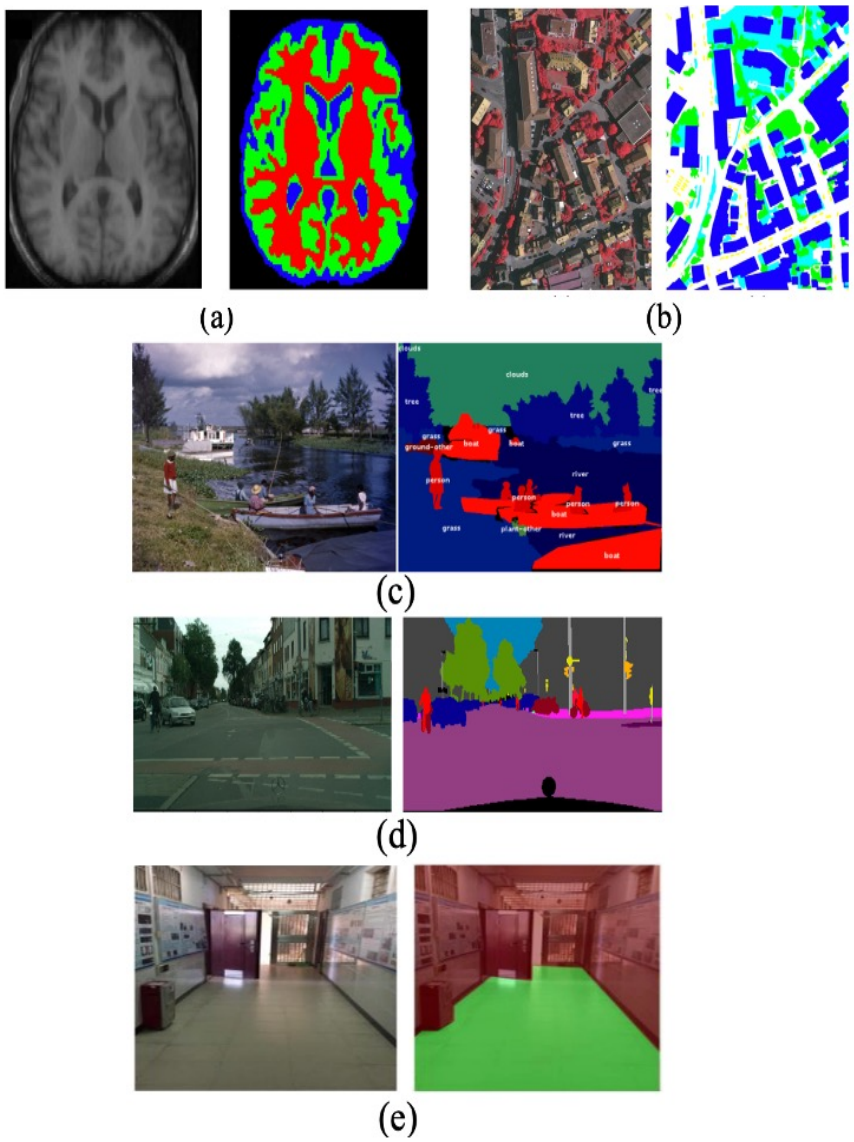


Figure 1.1: Examples of semantic segmentation Applications. (a) A slice of a 3D volume of a MR image from IBSR dataset and its segmentation result [Withey and Koles, 2008], (b) Example patch of the semantic object classification contest with true orthophoto and its ground truth [Song and Kim, 2020] https://www.isprs.org/education/benchmarks/UrbanSemLab/2d-sem-label-vaihingen.aspx, (c) Annotated image from the COCO-Stuff dataset with dense pixel-level annotations for stuff and things [Caesar et al., 2018], (d) a sample from The Cityscapes Dataset for Semantic Urban Scene Understanding [Cordts et al., 2016], (e) a Sample from an indoor dataset and accompanying labels for robot navigation [Yeboah et al., 2018]

spatial dimension through some pooling layers to capture all higher semantic information needed for segmentation, whilst decoder part gradually recovers the object details and spatial dimension to reconstruct the resolution lost by downsampling operation by combining multi-level feature maps from encoder. By using skip connections, each feature map of the decoder part receives the information from the correspondingly cropped feature map at the same level of the encoder part.

1.2 Motivation

Deep Convolutional Neural Networks (DCNNs) have led to significant breakthroughs in scene understanding. In the realm of semantic segmentation methods, the primary focus of enhancements has been on improving classification accuracy to achieve highly precise results. While 'performance' in this context includes not only accuracy but also efficiency and resource optimization, the quest for high performance is predominantly marked by the challenge of finding a balanced trade-off among these factors. Achieving a semantic segmentation method, that excels in accuracy while maintaining efficiency and optimal resource use remains an intricate challenge. The preliminary deep learning techniques mostly focus on the image classification task [Deng, 2012, Zhao et al., 2017, Krizhevsky et al., 2012], where image is analysed to assign a label from a set of pre-defined classes to an entire image. In contrast, semantic segmentation is the process of assigning a class label to each pixel in an image, where pixels that are close and have similar spectral characteristics are then grouped together into a segment that represents a specific object in the image. Semantic segmentation provides richer information, including object's boundary and shape, therefore, the semantic segmentation task requires a different deep neural network structure compared to the image classification task. It inherits all the problems from classification, such as appearance changes, scales and lighting conditions, cropping and occlusion. These problems become even more severe, because semantic segmentation must not only identify the semantic class, but also estimate its exact location. Fully convolutional-based neural networks [Long et al., 2015, Ronneberger et al., 2015] provide an end-to-end learning method for semantic segmentation, whilst avoiding the tremendous increasing of parameters, caused by reconciling classification to segmentation. Before the advent of fully convolutional methods, classification models were often repurposed for segmentation tasks. This adaptation involved treating each pixel or region as an independent classification problem, necessitating modifications to the models. Consequently, the number of parameters grew substantially, as each pixel or region required its own set of parameters. In contrast, the fully convolutional method addressed segmentation directly, using convolutional layers to capture spatial dependencies without the need for extensive modifications. This approach resulted in more parameter-efficient solutions, as the models were designed specifically for segmentation tasks from the outset.
Most recent improvements in both patch-wise or pixel-wise CNN-based methods for semantic segmentation were accomplished by increasing the network size [Simonyan and Zisserman, 2014, He et al., 2016], whereas deeper networks provoke large computational costs. It makes them unsuitable for real-time systems, that require low-

latency operations. Advanced Driver Assistance Systems (ADAS) are examples of these applications, due to their crucial need to take decisions in precise intervals. Therefore, improving semantic segmentation models towards achieving both high accuracy and low latency is of crucial importance.

Goal 1: In this thesis, we focus on an efficient semantic image segmentation task to fill this gap and propose a general concept based on the Convolutional Neural Network(CNN) to solve the semantic image segmentation task in a time and resource budget, applicable for real time systems.

While deep convolutional neural networks are strongly powerful to learn the local features for the pixel-level labelling task in an end-to-end way and perform well segmentation results, they have drawbacks to utilize global context information to model the interactions and correlation between the output variables directly. The label prediction by CNN-based method is done for each local neighbouring interdependently without including the information from surrounding labels. CNNs lack to encourage label agreement between similar pixels and to keep the spatial and appearance consistency of the labelling output, which lead to obtain a very coarse segmentation result. Furthermore, having larger receptive field of convolutional network along with max-pooling layers in CNNs reduce the chance of getting a fine segmentation output especially along the object boundary. Thus, simple feed forward CNNs may not be the perfect model for smoothed semantic segmentation and some further post processing step is needed. Several works have successfully combined the effectiveness of CNNs with the probabilistic graphical models such as CRFs to address the discussed issues.

Goal 2: For the reason stated above, a further goal of this thesis is to study the use of different CRF based techniques as a post processing step to move from a coarse to fine segmentation result. We investigate their effectiveness regarding their accuracy and time efficiency to keep our main goal for using in real time systems.

Despite of great success of fully supervised machine learning methods for solving various problems in computer vision, they suffer from having large amount of annotated training data from a specific context and a specific application. It leads to two difficulties. First, annotating data for pixel-wise classification tasks (such as semantic segmentation) is difficult and costly, especially for high-resolution images that contain millions of pixels. Second, supervised models can only perform at their maximum precision, when applied to the same or similar training data. To fulfill this requirement, we should have enough annotated data set to train models to make them robust for all real-world contexts in that particular application. Applying a pre-trained semantic segmentation model to an unseen domain, such as a new city, whose images are not presented in the training set, will not achieve satisfactory performance due to data set biases [Chen et al., 2017]. While semi-supervised semantic segmentation methods using partial annotations [Papandreou et al., 2015, Vezhnevets et al., 2012] alleviate this problem, they still require some pixel-level ground truth. To rectify mentioned

problems and adjust the methods for better generalization, recently, unsupervised techniques based on Generative Adversarial Network(GAN)[Goodfellow et al., 2020] have been growing as a powerful technique to improve the generalizability of deep learning methods. They can translate one domain to another domain and adapt data for use in a new domain and dispel the necessity of having costly labelled data.

Goal 3: This leads to the third goal of this thesis to exploit unsupervised learning for the semantic segmentation to completely strengthen our proposed method to be adaptable for domain with different underlying data distribution, while fulfilling the current performance requirement.

1.3 Contribution

This dissertation focusses on an efficient semantic image segmentation method to obtain the most accurate solution as possible for a restricted time budget, usable for real time systems. As the case study, all works presented here contribute to the research in modern ADAS, that should be accurate and computationally efficient to execute in real time on embedded platforms. This study, comprise semantic segmentation for pixel-wise classification of different objects in urban scene images mainly tested for road segmentation. All proposed methods constituted the state of the art, at the moment of completion. This work addresses different challenges which are summarized below:

Fully supervised Super pixel-based Convolutional Neural Network for Semantic Segmentation

As a first contribution, in this thesis, we proposed a novel approach to utilize the advantages of supervised CNNs for the task of semantic segmentation at the suitable computational effort. This method mainly differs from usual semantic segmentation methods in two aspects: first the input data model for the CNN network and second the simple CNN-network layering. The state-of-the-art convolutional neural networks for image segmentation are based on two different input data model: They are based on either patch-wise [Farabet et al., 2013, Ciresan et al., 2012, Ning et al., 2005] or pixel-wise dense classification [Long et al., 2015]. The most recent improvements in CNN's are profited by using above input data models and increasing the network size, which together require powerful GPUs. Since deeper networks cause large computational costs, they are mostly not suitable for real time systems. In Comparison to above state-of-the-art models, our method combines larger basic units "superpixels" with simple network structure, which tremendously reduces the input size. This strategy disassembles the pixel grid into super pixels forming the basic units for the classification task by CNN. The proposed CNN for super pixel-level image segmentation applies conventional supervised learning and uses training data with corresponding defined superpixel ground truth as prior information. Reducing the input to the superpixel domain allows the CNN's structure to stay small and efficient to compute, while keeping the advantage of convolutional layers.

- An existing approach [Achanta et al.,] is adopted to perform well-segmented superpixel segmentation over each single image, which have recently shown success as the fast and accurate bottom-up grouping of pixels method. Here, super pixels [Ren and Malik, 2003] are described by a spectral similarity and spatial proximity to enforce compactness and homogeneity within the super pixel.

- This thesis shows the incorporation of high-level declarative prior knowledge by hand-crafting features. We define for each super pixel a high dimensional feature descriptor, which extracts relevant informative superpixel characteristics in the image to facilitate the subsequent learning and generalization steps.

- The superpixels are assigned to corresponding positions of a regular grid structure extending across the image in order to create neighbourhood relations for convolutional purpose.

- This lattice together with the image descriptors are fed to a proposed convolutional neural network to classify the superpixels of the image according to semantic categories.

- Two cases from different datasets are used as examples to show that the proposed method is effective for road segmentation tasks. The strengths and the drawbacks of the proposed superpixel CNN based method are summarized through illustrative test and case studies. In particular, we conduct several comprehensive road segmentation tests by attentively change the size or homogeneity (regular or irregular) of superpixel and observe the effect on the outcomes of the CNN models in two perspectives named Birds-Eye view (BEV) and Image View or First-Person View (FPV). While, BEV provides a top-down perspective of a scene or environment, showing objects and their spatial relationships from an overhead viewpoint, Image-View represents the viewpoint of an observer or camera within the scene, capturing the view as perceived from the observer's position.

Fine-grained Super pixel-based Convolutional Neural Network for Semantic Segmentation using Conditional Random Field(CRF)

In the proposed SP-CNN method, the runtime profits from using few layers and irregular super pixels as basis for the input for the CNN rather than regular patch or full image, which tremendously reduces the input size. Although, this method achieved remarkable low computational time in both training and testing phases, the lower resolution of the super pixel domain yields naturally lower accuracy compared to high-cost state of the art methods. Furthermore, Convolutional Neural Networks (CNNs) have limitations, when it comes to capturing the direct dependency between output variables. In conventional CNN approaches, each label variable is predicted independently, which can compromise the achievement of a smooth segmentation.

- In the second contribution of this dissertation, we propose a model strategy to refine the classification result from super pixel grid to pixel grid using Conditional Random Fields (CRFs) [Lafferty et al., 2001]. CRFs can model global properties like object connectivity, geometric properties, and spatial relationship between objects.
- The key idea of CRF inference for semantic labelling is to formulate the label assignment problem as a probabilistic inference problem, that incorporates assumptions such as the label agreement between similar pixels or image regions. This work comprises two aspects for coupling local and global evidence. We combine the local image classification information extracted from CNN part with global information of neighbouring pixel relations to decide for an accurate pixel label.
- To make the semantic labelling assignment computationally feasible and effortless, the refinement procedure is limited to the inaccurate predicted super pixels labelling from the previous step along the object border.
- Three different CRF optimization techniques are investigated for the task of pixel-wise image segmentation, considering the labels as hidden states and solving the label prediction as a solution to the inference problem. Different inference methods are used to study the influence of computational gain and overall accuracy, allowing for robust and accurate statistical analysis of semantic segmentation. Experiments show that the final proposed refinement procedure can obtain comparable performance among the top-performing algorithms on the case study for road segmentation.

Un-supervised smooth Super pixel-based Convolutional Neural Network for Semantic Segmentation

The approach presented before based on a supervised CNN method and a probabilistic graphical model, like other fully supervised semantic segmentation methods, is suffering from the lack of a robust learning mechanism that is generalizable over different conditions. The previous approach, employing a combination of supervised CNN method and probabilistic graphical model, is subject to a common challenge shared by many fully supervised semantic segmentation methods. It lacks a robust learning mechanism capable of effectively adapting to diverse conditions and achieving generalizability. For humans, it is easy to distinguish between roads, sidewalls, grass, or existence of shadow on the road surface irrespective of lighting conditions, surroundings, and other variations. To achieve such a high degree of robustness to variations, a powerful and stable learning method is required. Contrary to the traditional methods, the supervised deep learning methods provide the potential ability to construct such a learning algorithm, removing a great deal of hand-engineering, while still retaining a large modelling capacity. However, they are highly dependent on the very large amount of training data set with high costs of manual annotation, that have equal underlying distribution with testing data. Whereas, having diversity between the training and testing data is common in the real world.

- Our next contribution in this dissertation is to rectify above problem and adjust the method for a better generalization. We propose an unsupervised technique based on Cycle Consistence Adversarial Network (CycleGAN) [Zhu et al., 2017] to exploit this intuition as a powerful technique to improve the generalizability of proposed super pixel based convolutional neural network model and enhance the super pixel semantic segmentation result.
- Primarily, we defined a 4 channel augmented input data model and demonstrate how our framework can process new imagery data in order to generate a real-looking of fine-grained samples.
- Secondly, we algorithmically and architecturally adapt the adversarial model framework to the new data model to handle the trade-off among the segmentation accuracy, memory resources and inference speed for large-scale image size. We experimentally show that better budget-aware segmentation results (in both timing and computational resource) are achieved by the proposed method comparing to other unsupervised methods for road segmentation as our case study.

1.4 Organization

The rest of the thesis is organized as follows:
Chapter 2 provides a detailed description of the technical background for the methods, we present in this thesis for the task of semantic image segmentation. It gives in-depth understanding of the notations and terminologies used in later chapters. Meanwhile, it gives an overview of Artificial Neural Networks (ANNs) and their fundamental parts with the focus on the Convolutional Neural Network (CNN) architectures, which are the main thread to the whole proposed approaches in this work, and provides a guideline to the following chapters.
The next three chapters discuss our novel techniques for semantic scene segmentation with the less computational efforts and time-budget applicable for real-time systems. All three proposed methods are applied to semantic road segmentation task. The models are trained on two public available data sets in autonomous driving and several experiments are conducted to show the strengths or drawbacks of the methods.
In Chapter 3, we introduce our novel technique incorporating superpixels with a CNN network for achieving a fast and computationally efficient semantic segmentation, while maintaining an acceptable level of accuracy. A simple CNN architecture with a new input data scheme based on super pixels integrated with the high-quality hand-crafted features is proposed in this chapter. The general concept presented here is used as the foundation for the rest of the two proposed methods.
In order to address the speed-accuracy trade-off discussed in Chapter 3, Chapter 4 introduces a novel approach aimed at improving model performance. The proposed method leverages graphical models to enhance accuracy without compromising efficiency. Experimental results on road segmentation demonstrate the effectiveness of this approach in achieving improved performance.

Chapter 5 focuses on un(semi)supervised method for coarse-to-fine semantic segmentation task to alleviate the fully supervised drawbacks and afford the Goal 3 of the contributions.
Chapter 6 summarize the contributions of the presented work and concludes the whole thesis. Moreover, it gives an outlook to the future works stemming from this research.

2 Technical Background

This chapter provides a comprehensive overview of the fundamental theory underlying deep neural networks. In Sections 2.1, 2.3 - 2.5, we delve into the architecture of key neural networks, their components, and the theoretical concepts and terms that form the basis of the methods proposed in this thesis. Additionally, we explore familiar learning schemes, optimization techniques, and diverse loss functions. By examining these topics, we establish a solid foundation for understanding the principles upon which our research builds.
Section 2.2 provides an overview of deep learning, outlining key training strategies and foundational principles of deep neural networks. Section 2.6 delves into Convolutional Neural Networks (CNNs), the cornerstone of machine learning methodologies in a plethora of semantic segmentation tasks, including the three contributions of this work. It offers a comprehensive introduction to CNNs, elucidating the architecture's layered structure and the pivotal roles these layers play.
Section 2.7 presents the essential concepts and taxonomy of image segmentation techniques, with a particular emphasis on deep learning-based semantic segmentation approaches, the core subject of this thesis. It will thoroughly examine prominent deep learning frameworks extensively applied to semantic segmentation, detailing their mechanisms and applications.

2.1 An Introduction to Artificial Neural Network (ANN)

Artificial Neural Networks (ANN) are a family of computational models inspired by human brain processes and designed to analyse and process the underlying relationship between sets of data. ANNs pave the way for scientific advances to achieve a high level of competence in solving many complex problems. The main components of an ANN are input/output, computational units (neurons), weights, biases, and activation functions, which are discussed below.

2.1.1 Artificial Neuron

Fundamental computational units in ANNs called neurons or nodes, wherethrough data and computations flow. Figure 2.1 shows the functionality of a neuron in an artificial neural network. Each neuron is characterized by its weight vector (W), bias (b) and activation function (f). They receive one or more inputs, that can come from either the raw data or from neurons of previous layer. Each input vector $\boldsymbol{x} = [x_1, \ldots, x_n]$ is associated with a weight parameter vector $W = [w_1, ..., w_n]$, that will be learnt through network training. Another input parameter is the bias (b), which is a scalar parameter and like the weights is learnable.

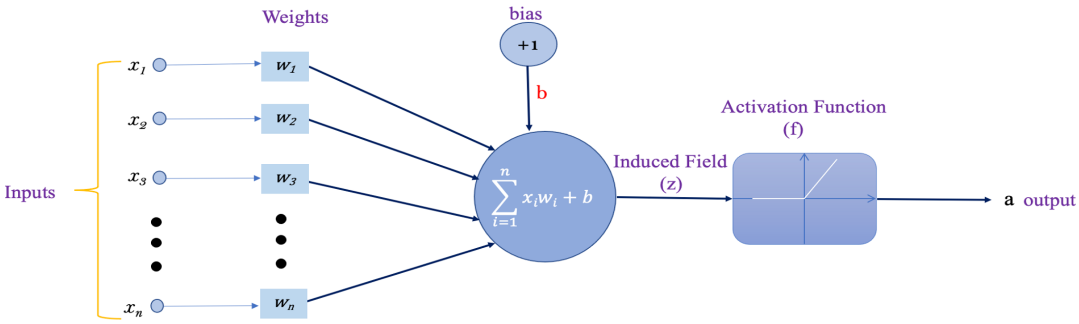


Figure 2.1: Neuron Functionality

A bias value is used to adjust the output along with the weighted sum of the inputs, to equilibrate the appropriate result with the given data. Each neuron performs a linear transformation on its input using the weights and biases. Then the result (z) is sent to an activation function, to decide whether a neuron should be activated or not. This means, it will decide whether the neuron's input to the network is important or not in the process of prediction using simple mathematical operation. This mathematical function can be stated as follows.

$$\begin{aligned} z &= \sum_{i=1}^{n} w_i x_i + b \\ \mathrm{a} &= f(z) \end{aligned} \tag{2.1}$$

2.1.2 Activation Functions

The activation function f in equation 2.1, as a non-linear transformation function, is a decision-making element, that signifies the decision boundary in the input domain by setting a threshold in the induced local field. Without an activation function, the output is a simple linear function, which always acts as a linear regression model. The result of stacking linear functions is also a linear function. However, most of the real world problems are complex and non-linear. The activation functions enable us to solve more complicated tasks. Several non-linear activation functions for neural networks are described in the literature [Goodfellow et al., 2016]. Theoretically, any differentiable function can be considered as an activation function. In practice, only few "well-behaved" (bounded, monotonically increasing, and differentiable) activation functions are used. Some of the popular activation functions are highlighted in Table 2.1. Generally, different nodes in the same or different layers of the neural network may have different activation functions. Nevertheless, almost all the networks use the same activation functions specially for the nodes in the same layer. Moreover, choosing the activation function strongly depends on the task to be solved or the better way of speaking, what our network is trying to learn.

Name	Function f	Derivative of f: f'	Range
Identity	$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Sigmoid	$f(x) = \frac{1}{1 + exp(-x)}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
Gaussian	$f(x) = exp(-x^2)$	$f'(x) = -2x\, exp(-x^2)$	$(0, \infty)$
Thanh	$f(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)}$	$f'(x) = 1 - [f(x)]^2$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$	$f'(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$	$(0, \infty)$
LReLU	$f(x) = \begin{cases} 0.01x, & x \leq 0 \\ x, & x > 0 \end{cases}$	$f'(x) = \begin{cases} 0.01, & x \leq 0 \\ 1, & x > 0 \end{cases}$	$(0, \infty)$
PReLU	$f(x) = \begin{cases} \alpha x, & x \leq 0 \\ x, & x > 0 \end{cases}$	$f'(x) = \begin{cases} \alpha, & x \leq 0 \\ 1, & x > 0 \end{cases}$	$(0, \infty)$

Table 2.1: Common activation functions used in neural networks [Goodfellow et al., 2016, Aggarwal et al., 2018, He et al., 2015a]. The activation functions include Rectified Linear Unit (ReLU), which sets negative values to zero, Leaky ReLU (LReLU), allowing a small, non-zero gradient when the unit is inactive, and Parametric ReLU (PReLU), where the leakage coefficient becomes a learnable parameter along with other network parameters.

2.2 Understanding Deep Neural Network(DNN)

The way that processing elements (Neurons) in ANN are connected to each other is very essential in ANNs. There are different types of architectures for ANNs such as, single layer feed-forward, multi-layer feed-forward, Recurrent,etc. The simplest form, a feed-forward neural network, also known as a perceptron consists of three main layers of neurons.The first layer, called input layer is composed of n neurons, arranged as a n-dimensional vector, the hidden layer (could be more than 1 layer) is composed of m neurons, arranged as a m-dimensional vector, and the output layer. More generally, an ANN with multiple consecutive hidden layers is called Deep Neural Network (DNN) [Hinton et al., 2006]. They have become the most powerful solutions to complex tasks like Natural Language Processing, Computer Vision, Speech Synthesis. The hidden layers are iteratively breaking down the input into valuable information,

leaving out the redundant information and extract the important patterns, that are distinctive to a particular object. Relatively large number of layers in deep networks allows the network to automatically learn more complex patterns and relationships exist in the data.

2.2.1 Major Architectures of Deep Neural Networks

There are three different types of Deep Neural Networks, which are popularly used:

1. **Multi-Layer Perceptrons (MLP)**: Multi-Layer Perceptrons (MLPs) [Rosenblatt, 1958, Rumelhart et al., 1986] are a basic yet powerful type of deep neural network with a feed-forward architecture (see section 2.4.1). They consist of an input layer, multiple hidden layers, and an output layer. MLPs operate by taking an input vector and transforming it through these layers to produce an output, which serves as the network's prediction. The model learns by adjusting its parameters to reduce the difference between its predictions and the actual data. MLPs are versatile and can be applied to a wide range of tasks, from simple regression to complex classification problems.

2. ***Convolutional Neural Networks (CNN):*** A Convolutional Neural Network (CNN) [Widrow, 1962, Fukushima, 1980, LeCun et al., 1998] is another type of deep neural networks with spatially bounded parameters. CNNs are mainly used in Computer Vision or image processing for doing a specific task like image classification, object recognition, and semantic segmentation.

3. ***Recurrent Neural Networks (RNN):*** If feed-forward neural networks are extended to have feedback connections, they are called Recurrent Neural Networks [Elman, 1990, Hochreiter, 1997]. Comparison to other feed-forward networks, they are able to send information over time-steps. From a sequence of input vectors, RNN takes each vector at a time and model them. This ability allows the network to preserve state, while each input is modelled.

2.3 Broad strategies for training ML models

Learning approaches differ based on the interaction between the learner and the environment, and the way of earning skills or knowledge obtained by synthesizing useful concepts from historical data. It is a goal-guided process for the improvement of the system's behaviour due to experience and prior knowledge. According to the learning process and the type of output or the problem it attempts to solve, they are largely divided to four main recognized categories [Goodfellow et al., 2016, LeCun et al., 2015]:

- Supervised
- Unsupervised

- semi-supervised
- Reinforcement

In supervised learning, the objective is to find a general rule that maps inputs to ground truth labels. Here, a learning algorithm analyses the training data samples and produces a derived function, which will later be used for mapping new examples. In this learning method, labels have to be available during training.
Unsupervised machine learning, in contrast to supervised, does not need any labelled data. Here, the model learns to use techniques on the input data to mine for rules, detect patterns, and cluster(group) the data points, that help to obtain meaningful insights within datasets without reference or labels to be known.
Semi-supervised learning as a Hybrid Learning approach combines supervised and unsupervised learning, in which the training data contains a large number of unlabelled examples and few labelled ones. Semi-supervised learning attempts to improve the accuracy of supervised learning by exploiting information from cheap and abundant unlabelled data.
In Reinforcement Learning, machines continuously interacts with its environment in an iterative mode to take a suitable action to maximize reward within a particular context. The system evaluates its performance based on the feedback responses and reacts accordingly.
Training neural networks together with acquiring annotated datasets are cumbersome and computationally expensive. To overcome this problem, various techniques of transfer learning are proposed to leverage from pre-trained models as the starting point for training, instead of training a neural network from scratch. Transfer learning, reuses the model and data gained, while solving one problem and applies it to a new related task. Transfer learning methods can be categorized in inductive and transductive [Ribani and Marengoni, 2019]. Inductive transfer learning is a powerful technique that enables the transfer of knowledge from one domain to another, leading to improved learning and performance. It involves utilizing labelled data from both the source and target domains during training. The process begins with pre-training a model on a large dataset or alternative data sources like simulation data or generative models. Subsequently, the model is fine-tuned using a smaller target dataset that has limited labelled examples. By leveraging the learned features and knowledge from the source domain, the model can benefit from the general representations that are relevant to both domains. This approach is particularly effective when the shared features between the domains are more general rather than specific to the source domain alone. Examples of inductive transfer learning include multi-task learning and self-taught learning, which further demonstrate the potential and versatility of this approach.
Transductive transfer learning refers to as domain adaption, can also be used to transfer knowledge from a source domain to a target domain, where no labelled target data is available, but the unlabelled data on the target domain can be seen during training. Domain adaption in deep neural networks is a very active research area and several

techniques in deep neural network for domain adaption are investigated in [Wang and Deng, 2018].

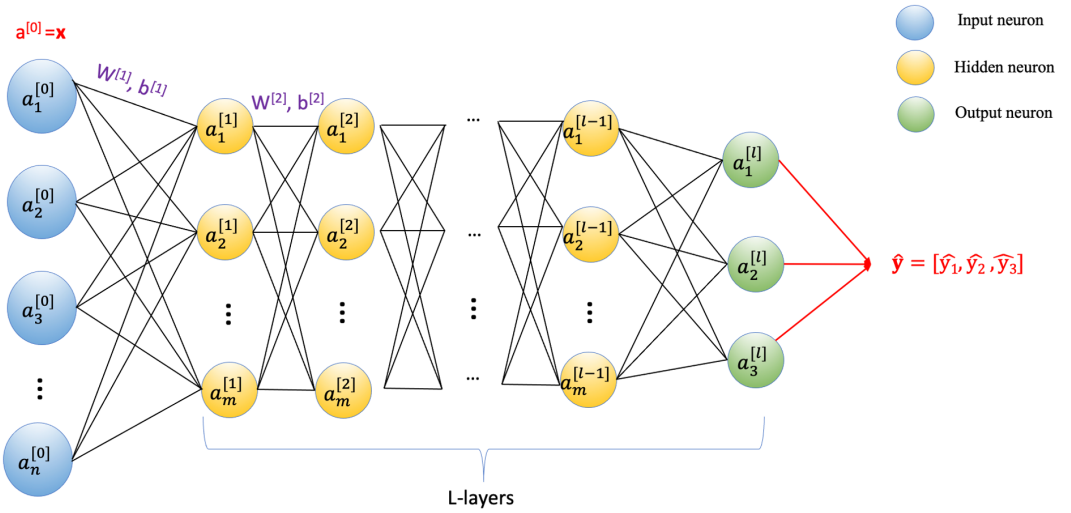


Figure 2.2: Scheme of a Multi-Layer Perceptron(MLP) network with n input features, m hidden neurons per layer, L layers and three outputs. The diagram exemplifies the forward propagation process, highlighting how data flows from the input layer through the hidden layers to produce the final output.

2.4 Training Deep Neural Networks

Training deep neural networks involves a two-phase process, encompassing the forward pass and the backward pass, utilizing the error back-propagation algorithm. During the forward pass, computation progresses layer by layer from the input towards the output, applying a mapping function and activating each layer's neurons non-linearly. Conversely, the backward pass involves computing the gradients of the loss function with respect to each weight, moving from the output back to the input layer. This bidirectional flow, known as forward and backward propagation, is fundamental to approximating the mapping functions central to the operation of contemporary deep learning models.

2.4.1 Forward Propagation

In the realm of feed-forward networks, the architecture is designed to allow information to move in a single direction from the input layer to the output layer, navigating through various hidden layers. The Multilayer Perceptron (MLP), a prime example of such networks, is recognized for its fully interconnected design. Every neuron in a

layer is connected to each neuron in the next layer, with distinct biases and weight sets facilitating the flow of data. Figure 2.2 illustrates the forward propagation in an MLP, tracing the input's journey through multiple hidden layers to the output. This visualization underscores the MLP's linear progression, where each layer's output becomes the input of the next layer, culminating in the final output. This structured data flow allows the MLP to decode and learn intricate patterns within the input data, making it a versatile tool for tasks ranging from classification to numerical approximation.

Training a Multilayer Perceptron (MLP) involves mapping an input vector $\boldsymbol{x} = [x_1, \ldots, x_n]$, where each x_i is an input feature, through multiple L layers to produce a predicted output $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_C]$, where C signifies the number of output neurons. The transformation denoted by $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; W, B)$ incorporates a series of learnable weight matrices $W = [W^{[1]}, \ldots, W^{[L]}]$ and bias vectors B = $[b^{[1]}, \ldots, b^{[L]}]$, with $W^{[l]}$ being the weight matrix that connects the $(l-1)$-th layer to the l-th layer, and $b^{[l]}$ representing the corresponding bias vector for the l-th layer. Each layer, from the input to the output, contributes to this mapping by performing a weighted summation of its inputs, adding the bias vector $b^{[l]}$, and applying a non-linear activation function to produce activations $a^{[l]}$ for the next layer. The final layer, or the output layer, outputs the prediction $\hat{\boldsymbol{y}}$, employing an activation function suited to the task at hand. The objective of training is to fine-tune the weights W and biases B, so that $\hat{\boldsymbol{y}}$ closely approximates the actual outputs $\boldsymbol{y} = [y_1, \ldots, y_C]$, utilizing back-propagation and optimization techniques to minimize the loss function that measures the difference between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ (see section 2.4.2).

In initial step of the training procedure, all weights in the network are stochastically initialized (see section 2.5.2). The general statement of the equation 2.1 for j^{th} node of the l^{th} layer has the following equations:

$$\begin{aligned} z_j^{[l]} &= \sum_{k=1}^{m} w_{jk}^{[l]}\, a_k^{[l-1]} + b_j^{[l]} \\ a_j^{[l]} &= f^{[l]}(z_j^{[l]}) \end{aligned} \tag{2.2}$$

Here, The superscript indexes indicate the number of the corresponding hidden layer and the subscript indexes shows the number of node. $w_{jk}^{[l]}$ refers to the weight associated with the neuron j in the layer l with the neuron k from layer $l-1$. For every layer in the network, certain parameters associated with following notations are considered:

- $\mathrm{n}^{[l]}$: The number of neurons in the layer l,
- $\mathrm{W}^{[l]}$: The weight matrix associated with the layers l and $l-1$ with the size of $(n^{[l]} \times m^{[l-1]})$. w_{jk}^{l} refers to the weight associated with the neuron j in the layer l with the neuron k from layer $l-1$,

- $\mathrm{b}^{[l]}$: The vector of biases associated with the layer l of the size $(n^{[l]} \times 1)$,
- $\mathrm{a}^{[l]}$: The vector of activations of the neurons in the layer l of the size $(n^{[l]} \times 1)$,
- $\mathrm{z}^{[l]}$: The vector of the sum of the weighted output of the neurons in the layer l and the size $(n^{[l]} \times 1)$,
- $\mathrm{f}^{[l]}$: The activation function applied to the the neurons in the layer l with $a^{[l]} = f^{[l]}(z^{[l]})$.

To calculate $a^{[l]}$ for every layer l in the network, we should calculate an intermediate vector $z^{[l]}$. If $\boldsymbol{x}$ is the input vector to the neural network ($a^{[0]} = \boldsymbol{x}$), then $z^{[l]}$ is obtained from the following equation:

$$a^{[l]} = f^{[l]}(z^{[l]}) = f^{[l]}(w^{[l]} \cdot \ a^{[l-1]} + b^{[l]}) \tag{2.3}$$

Here, the activation of the final layer represents the neural network's predicted output $\hat{\boldsymbol{y}}$.

2.4.2 Back-propagation

Through backward-pass, also known as back-propagation phase, the network learns how far its predicted output is from the right answer, and then injects the error committed from this prediction (forward) phase back into the network to update its parameters (weights (w) and biases (b)) on the next iteration. This process leads to the actual output getting closer to the target output, and a better network learning [Rumelhart et al., 1986] and more specifically by [Bishop and Nasrabadi, 2006, pages 240-245].
The learning process begins with the forward propagation phase, where the input features of a single sample, $\boldsymbol{x} = [x_1, x_2, ..., x_n]$, are fed into the network. This step involves processing the data from left to right to predict the network's output, represented as $\hat{\boldsymbol{y}}$. During this phase, the true value $\boldsymbol{y}$ is already established. The second step is to determine how "wrong" the guess of the network is. For this purpose, a cost function $\mathcal{C}$ is used to calculate the error scoring (loss) between the forecast $\hat{\boldsymbol{y}}$ and actual value $\boldsymbol{y}$. In the context of a single sample, a commonly utilized cost function can be defined as:

$$\mathcal{C} = \frac{1}{2C} \sum_{i=1}^{C} (\hat{y}_i - y_i)^2 \tag{2.4}$$

Where C indicating the dimensionality of the output space, which could correspond to the number of classes in a classification setting or output variables in regression.
The training goal is to find a set of parameters (weights), that will minimize the cost function. In the third step, the weights are updated based on the calculated error.

Finally, the forward and backward passes repeat until custom epoch count, where the error reaches to a minimum value.
The back-propagation algorithm leverages the gradients of the cost function relative to the network's parameters to guide the updates, thereby systematically improving model performance. A cornerstone technique for adjusting the weights to minimize the cost function is Gradient Descent (GD) [Goodfellow et al., 2016, page 83]. GD iteratively updates each weight by moving it in the opposite direction of its cost function's gradient, effectively steering the parameters towards the minimum of the cost function. For a network with L layers, the update rules for the weight $w_{jk}^{[l]}$ connecting the k-th neuron in layer $l-1$ to the j-th neuron in layer l and the bias $b_j^{[l]}$ associated with the j-th neuron in layer l, during iteration t, are given by:

$$\begin{aligned} w_{jk}^{[l]}(t) &= w_{jk}^{[l]}(t-1) - \eta \frac{\partial \mathcal{C}}{\partial w_{jk}^{[l]}(t-1)} \\ b_j^{[l]}(t) &= b_j^{[l]}(t-1) - \eta \frac{\partial \mathcal{C}}{\partial b_j^{[l]}(t-1)} \end{aligned} \tag{2.5}$$

Where η is the learning rate, and $\frac{\partial \mathcal{C}}{\partial w_{jk}^{[l]}}$ is the gradient of the loss function $\mathcal{C}$ with respect to the weight $w_{jk}^{[l]}$ and $\frac{\partial \mathcal{C}}{\partial b_j^{[l]}}$ denotes the gradient of the loss function $\mathcal{C}$ with respect to the bias $b_j^{[l]}$. To apply back-propagation, we compute the partial derivatives $\frac{\partial \mathcal{C}}{\partial w_{jk}^{[l]}}$ and $\frac{\partial \mathcal{C}}{\partial b_j^{[l]}}$, which quantify how changes in weights and biases affect the loss. For a weight w_{jk} in a simple network with a single layer and a bias $b = 1$, the gradient of the loss with respect to w_{jk} is:

$$\frac{\partial \mathcal{C}}{\partial w_{jk}^{[1]}} = \frac{\partial \mathcal{C}}{\partial a_j^{[1]}} \cdot \frac{\partial a_j^{[1]}}{\partial z_j^{[1]}} \cdot \frac{\partial z_j^{[1]}}{\partial w_{jk}^{[1]}} \tag{2.6}$$

Here, $a_j^{[1]}$ denotes the activation of the j-th neuron in layer 1, and $z_j^{[1]}$ represents the weighted input to this neuron, calculated as $z_j^{[1]} = \sum_k w_{jk}^{[1]} x_k + b$, with x_k being the input from layer 0 and b being the bias. Given that $\frac{\partial z_j^{[1]}}{\partial w_{jk}^{[1]}}$ equals x_k (the input from the previous layer), the formula simplifies to:

$$\frac{\partial \mathcal{C}}{\partial w_{jk}^{[1]}} = \frac{\partial \mathcal{C}}{\partial a_j^{[1]}} \cdot \frac{\partial a_j^{[1]}}{\partial z_j^{[1]}} \cdot x_k \tag{2.7}$$

To extend this concept to a multi-layer perceptron (MLP) with L layers, we define the error term $\delta_j^{[l]}$ for the j-th neuron in the l-th layer as the derivative of the cost function with respect to the neuron's weighted input $z_j^{[l]}$:

$$\delta_j^{[l]} \equiv \frac{\partial \mathcal{C}}{\partial z_j^{[l]}} \tag{2.8}$$

Considering that $\frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}}$ is the activation $a_k^{[l-1]}$ from the previous layer, the gradient of the cost function with respect to the weight can be expressed using the error term:

$$\frac{\partial \mathcal{C}}{\partial w_{jk}^{[l]}} = a_k^{[l-1]} \cdot \delta_j^{[l]} \tag{2.9}$$

The value of δ for each neuron is computed by propagating these errors backward from the output layer. This methodical adjustment of weights and biases, based on the back-propagation of errors, allows for the gradual improvement of the model's performance by minimizing the discrepancy between the predicted and actual outputs. For the entire network, the derivatives of the cost function with respect to the weights and biases in any layer l are calculated using the chain rule and are based on the error terms of the neurons:

$$\begin{aligned}
\frac{\partial \mathcal{C}}{\partial W^{[l]}} &= \frac{\partial \mathcal{C}}{\partial a^{[L]}} \cdot \left(\prod_{i=l+1}^{L} \frac{\partial a^{[i]}}{\partial z^{[i]}} \cdot \frac{\partial z^{[i]}}{\partial a^{[i-1]}} \right) \cdot \frac{\partial a^{[l]}}{\partial z^{[l]}} \cdot \frac{\partial z^{[l]}}{\partial W^{[l]}} \\
\frac{\partial \mathcal{C}}{\partial b^{[l]}} &= \frac{\partial \mathcal{C}}{\partial a^{[L]}} \cdot \left(\prod_{i=l+1}^{L} \frac{\partial a^{[i]}}{\partial z^{[i]}} \cdot \frac{\partial z^{[i]}}{\partial a^{[i-1]}} \right) \cdot \frac{\partial a^{[l]}}{\partial z^{[l]}}
\end{aligned} \tag{2.10}$$

These gradients are computed recursively from the output layer back to the input layer, enabling the network to adjust its weights and biases to minimize the loss function and improve the model's accuracy by reducing the discrepancy between the predicted and actual outputs.

2.5 Common Principles of Deep Networks

2.5.1 Vanishing or Exploding Gradient

The vanishing gradient problem [Hochreiter, 1991] is one of the key challenge in training deep neural networks, that can be happen either by improper weight initialization (section 2.5.2), or through updating weights using back-propagation (section 2.4.2), where the derivatives of the network is found with the chain rule by moving layer by layer from the final layer to the initial one. When n hidden layers use an activation like the sigmoid function, the gradient decreases exponentially as we propagate down to the initial layers. That leads weights and biases will not be updated effectively with each training session. One of the simple solution to this problem is to replace the activation function of the network by ReLU.
In contrast, exploding gradient[Goodfellow et al., 2016] is a problem where large error gradients accumulate and resulting in large weight updates to neural network model during training. This leads the model being unstable and unable to learn from the training data. At an extreme, the values of weights can become so large as to overflow and result in NaN(Not a Number) values, that can no longer be updated. Exploding gradient problem can be solved by redesigning the network model, using rectified linear activation, or weight regularization (section 2.5.3). Main solution to the exploding gradient problem is to prevent the gradients from getting too large by applying a process called gradient clipping, which imposes a predefined threshold on each gradient. Gradient clipping ensures that gradients move in the same direction but with shorter lengths.
In general, to prevent the vanishing or exploding in the gradients of the network's activations, the variance of the activations should not change across every layer. However, the distribution of activations can shift and scale in a forward pass, leading to a deterioration in weight distribution and consequently slowing down training and triggering side effects between layers.

2.5.2 Weight Initialization

Weight initialization is a method, where the weights of the neural network are set to small random values. This values define the starting point of optimization algorithm (learning or training) of the neural network model. The proper weight initialization is essential for network to learn well. The selection of too large or too small weights leads to the layer activation outputs exploding or vanishing through a deep neural network and the network will take longer to converge, if it is even able to do so at all. Weights in neural networks are mostly initialized with small and different random numbers to break symmetry as much as possible. This causes the activations to be zero-centred and have specific variance. Accordingly, they could keep specific distribution over multiple layers and each neuron can learn its distinct feature. A state-of-the-art weight initialization approach, that uses the Sigmoid or Tanh activation function is called "Glorot" or "Xavier" initialization [Glorot and Bengio, 2010], where scaled ran-

dom uniform or Gaussian distribution is used for the initial weights W of each layer. In [Glorot and Bengio, 2010] a variance $\nu(W)$ of the weight initialization distribution is proposed based on the number of neurons n_{in} in the previous and the number of neurons n_{out} in the current layer. The primary objective of Xavier initialization is to maintain uniform variances for both activations and gradients across all network layers. Such a consistency is crucial for preventing gradients from vanishing or exploding during training, thereby facilitating a smoother and more effective optimization process. It aims to ensure, that the variances of the layer inputs (pre-activations) and the layer outputs (post-activations) remain balanced, i.e $\nu(a^{[l]}) \approx \nu(z^{[l]})$. The method achieves this by scaling the variance of the initial weights $\nu(W)$ according to a compromise between the number of neurons in the connecting layers. Specifically, when n_{in} equals n_{out}, the variance scaling perfectly balances the forward and backward propagation needs. In general, to accommodate layers where n_{in} and n_{out} differ, Xavier initialization averages these two values, as outlined in the following equation:

$$\nu(W) = \frac{2}{n_{in} + n_{out}} \tag{2.11}$$

This approach ensures a uniform initial distribution of neuron outputs across the network, which has been shown to significantly enhance the convergence rate during training.
Above assumption is invalid for ReLU and PReLU (table 2.1). In the specific case of the neural networks with ReLU, half of the results are truncated. To overcome this, He et al. [He et al., 2015a] proposed to only rely on the number of neurons in the previous layer. This standard approach for initialization of the weights of neural network layers that use the rectified linear (ReLU) activation function is called "He" initialization. The He initialization method is calculated as a random number with a Gaussian probability distribution with a mean of 0 and a standard deviation of $\sigma_{\text{He}} = \sqrt{\frac{2}{n_{in}}}$.

$$\nu(W) = \frac{2}{n_{in}} \tag{2.12}$$

2.5.3 Model Regularization

State-of-the-art neural networks have millions of parameters with many degrees of freedom. To improve the model prediction of the networks, selecting and tuning the preferred level of the model complexity is an unavoidable and important step [Bishop and Nasrabadi, 2006, chapter 5.5]. The Poor model selection can lead to diminishing or exploding gradients that are accumulated by the chain rule of derivatives. Taking into account the bias-variance trade-off prevents the model from under-fitting or over-fitting. Under-fitting occurs due to oversimplification of the model (high bias). This problem mainly occurs either by improper weight initialization or by using some

bounded activation functions such as Sigmoid or Tanh (section 2.1.2). Bounded functions also have a bounded gradient, which can exponentially decrease the gradient, when used in every layer. Over-fitting is caused by over-complication of the model (high variance). It happens when a model learns the details and noise in the training data, which negatively impacts the performance of the model on new data. By using regularization technique, over-fitting can be prevented. It discourages the learning of a model of both high complexity and flexibility, by adding an additional penalty term in the error function. The two common regularization techniques are L_1 and L_2 :

$$\begin{aligned} L_1 &= \lambda \sum_{i=1}^{n} \mid w_i \mid \\ L_2 &= \lambda \sum_{i=1}^{n} (w_i)^2 \end{aligned} \tag{2.13}$$

where n is the number of weights w_i, $|w_i|$ is the absolute value of each weight, w_i^2 is the square of each weight and the parameter $\lambda \in \mathrm{R}$, as the penalize value, controls the influence of the regularization. L_1 regularization, also known as Lasso regularization, incorporates the absolute value of the coefficients into the loss function as a penalty term. On the other hand, L_2 regularization, also referred to as ridge regression, adds the squared magnitude of the coefficients as the penalty term. Regularization with L_1 promotes sparsity in the model by driving many coefficients to zero, and L_2 favours a Gaussian distribution of the weights [Hastie et al., 2009, pages 610-611].
Weight decay [Goodfellow et al., 2016, pages 230-237] is an effective regularization technique to mitigate over-fitting by subtly modifying the weight update rule in gradient descent. Unlike direct inclusion of L_1 or L_2 regularization terms in the loss function, weight decay incorporates a shrinkage factor, governed by the parameter λ, into the weight update process (see equation 2.5). This approach adjusts the weights by not only considering the gradient of the cost function, but also by scaling down the weights from the previous iteration, as outlined in the following equation:

$$w_{jk}^{[l]}(t) = w_{jk}^{[l]}(t-1) - \eta \left(\frac{\partial \mathcal{C}}{\partial w_{jk}^{[l]}(t-1)} + \lambda w_{jk}^{[l]}(t-1) \right) \tag{2.14}$$

This formula ensures a gradual reduction in the magnitude of the weights with each training iteration. By systematically decreasing the weights, weight decay helps in keeping the model parameters modest, thereby steering the learning process towards simpler models that generalize better to unseen data. This technique not only helps in controlling over-fitting but also contributes to a more stable and robust learning dynamics.
Dropout is another technique to prevent over-fitting in neural network [Goodfellow et al., 2016, page 258]. During training, some number of layer outputs are randomly ignored by setting their value to 0. It makes the training process noisy and prevent

the situations, where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust. In general, dropout increases the training time, however improves the performance of the network.

2.5.4 Batch Normalization

Batch normalization [Ioffe and Szegedy, 2015], or batch-norm is a technique to increase the stability and performance of neural network training. It normalizes the activations of intermediate layers with zero mean and a standard deviation of 1, which makes the training faster and over-fitting smaller.
As outlined in section 2.4.1, $z_j^{[l]}$ represents the aggregated weighted inputs (pre-activation output) for the j-th neuron in layer l. The symbol $\hat{z}_j^{[l]}$ is introduced to represent these inputs in their normalized form before applying the activation function. During the batch normalization process for a batch containing M samples, the empirical mean $\mu_j^{[l]}$ and variance $\nu_j^{[l]}$ are computed for each neuron j across the batch. For the i-th sample in the batch, the normalization process for neuron j within layer l encompasses these steps:

$$\mu_j^{[l]} = \frac{1}{M} \sum_{i=1}^{M} z_j^{[l](i)} \tag{2.15}$$

$$\nu_j^{[l]} = \frac{1}{M} \sum_{i=1}^{M} (z_j^{[l](i)} - \mu_j^{[l]})^2 \tag{2.16}$$

$$z_{\text{norm}j}^{[l](i)} = \frac{z_j^{[l](i)} - \mu_j^{[l]}}{\sqrt{\nu_j^{[l]} + \epsilon}} \tag{2.17}$$

In the normalization equation, ϵ is a small constant added to the variance $\nu_j^{[l]}$ to ensure numerical stability, by preventing division by zero. Following this adjustment, the activations are normalized to produce $z_{\text{norm}j}^{[l](i)}$. This step standardizes the activations to have zero mean and unit variance for each neuron across the batch, setting the stage for subsequent scaling and shifting.

$$\hat{z}_j^{[l](i)} = \gamma \cdot z_{\text{norm}j}^{[l](i)} + \beta \tag{2.18}$$

The parameters γ and β are learnable scale and shift factors for each neuron j in layer l, enabling the network to adjust the normalization effect if necessary. The normalized and adjusted output $\hat{z}_j^{[l](i)}$ then undergoes activation through the layer

activation function, $f^{[l]}$, resulting in the final activation $a_j^{[l](i)}$ for the i-th sample and j-th neuron at layer l:

$$a_j^{[l](i)} = f^{[l]}(\hat{z}_j^{[l](i)}) \tag{2.19}$$

This process, ensures that each layer inputs are standardized, promoting more consistent training dynamics and improving the model's learning capability. By ensuring that each layer receives inputs with consistent statistical properties, Batch Normalization helps in stabilizing the gradient flow through the network, making it possible to employ higher learning rates and accelerating convergence.

2.5.5 Loss Functions

A deep neural network learns how to map a set of inputs to a set of outputs by choosing the correct weights. Finding the perfect weights for a neural network is almost not possible. Instead, the model tries to navigate the space of possible sets of weights in order to make enough good predictions. The function used to evaluate a candidate solution called loss or objective function [Aggarwal et al., 2018].
Several training loss functions are used in Computer Vision. *Cross-Entropy* (CE) and *Mean Squared Error* (MSE) are the main two ones. For classification tasks, the cross-entropy is commonly used, which is defined for discrete events, where outputs (classes) are usually encoded in an indicator vector (one-hot vector) and the output of the network are interpreted as a probability distribution across the classes. Given a set of predicted probabilities, denoted as $\hat{\boldsymbol{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_C]$, where $\hat{y}_i$ represents the predicted probability for class i and C is the total number of classes, and the true labels represented as one-hot encoded vectors $\boldsymbol{y} = [y_1, y_2, ..., y_C]$, where $y_i = 1$ if the sample belongs to class i and $y_i = 0$ otherwise, the Cross-Entropy loss $\mathrm{L_{CE}}$ for total N samples is calculated as:

$$L_{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C}[y_{j,i}\log(\hat{y}_{j,i})] \tag{2.20}$$

In this formula, $\log(\cdot)$ denotes the natural logarithm. The term $y_{j,i}$ indicates if the j^{th} sample is in class i, while $\hat{y}_{j,i}$ represents the model's predicted probability, that the j^{th} sample falls into class i. The loss is computed by taking the negative logarithm of the predicted probability for the true class label and summing it over all classes.

Another common objective function is known as *Mean Squared Error* (MSE). It measures the average squared difference (known as euclidean distance) between the predicted value and the true label or target for each sample, and then takes the average across all samples. Traditionally, MSE is employed in single-output regression scenarios, where the goal is to predict a single continuous variable. However, in the realm of multi-output task(classification or regression), where the objective extends to pre-

dicting multiple variables from the same set of inputs, the MSE can be adapted to accommodate this complexity too. This adaptation involves summing over multiple continuous outputs (or classes) per sample. The formula for MSE loss (L_{MSE}) can be presented as following:

$$L_{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} (\hat{y}_{i,j} - y_{i,j})^2 \tag{2.21}$$

Squaring the differences has the effect of penalizing larger deviations more than smaller ones, as the squared terms amplify the differences. Therefore MSE loss is sensitive to outliers.
The concept of *Softmax loss*, also known as categorical cross-entropy loss, is pivotal in multi-class classification tasks, particularly when paired with the softmax activation function in the output layer. This loss function quantifies the disparity between the predicted class probabilities and the actual labels. To compute Softmax loss, we initially apply the softmax function to the logits or predicted scores. For a problem involving C classes, the softmax function serves as an extension of the sigmoid function, transforming the output vector $z^{[L]}$ of the final layer into a probability distribution, where each element's value ranges between $[0, 1]$, and the sum of all elements equals 1. The softmax function for the i-th sample and j-th class is defined as:

$$\text{Softmax}(z_{i,j}^{[L]}) = \hat{y}_{i,j} = \frac{e^{z_{i,j}^{[L]}}}{\sum_{k=1}^{C} e^{z_{i,k}^{[L]}}} \tag{2.22}$$

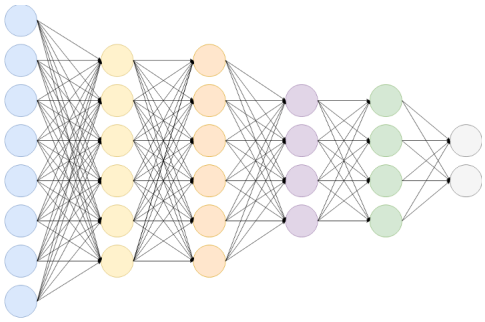
Here, $\hat{y}_{i,j}$ denotes the predicted probability, that the i-th sample is associated with class j, calculated by normalizing the exponential of the logit $z_{i,j}^{[L]}$ across all classes k for that sample, ensuring that the sum of probabilities for all classes equals 1. Subsequently, the Softmax loss (L_{Softmax}) is computed as:

$$L_{Softmax}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \log(\hat{y}_{i,j}) \tag{2.23}$$

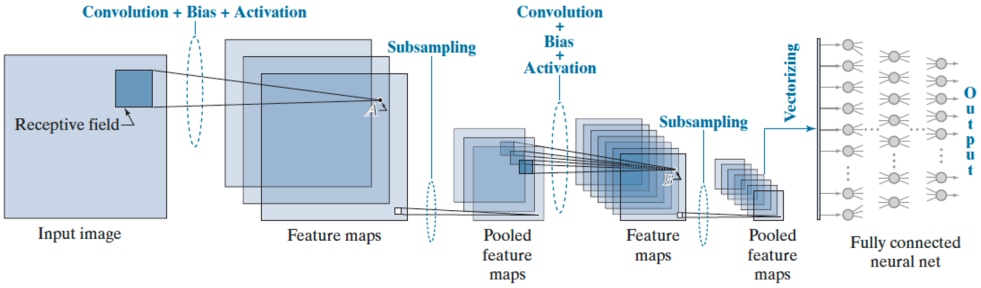
In this formula, $y_{i,j}$ acts as a binary indicator, being 1 if class j is the correct classification for the i-th sample, and 0 otherwise. The key distinction between Cross Entropy and Softmax Loss lies in the derivation of the predicted probabilities $\hat{y}_{i,j}$. While Softmax Loss specifically employs the softmax function to convert the model's logits into probabilities before applying the cross-entropy formula, Cross Entropy Loss can be applied in contexts, where the predicted probabilities are derived through different means, not limited to the softmax function.

The loss function can also be learned along with the network instead of explicitly formulating an objective function, by using Adversarial training. In [Goodfellow et al.,

2020] is proposed a combination of a generator and discriminator network known as Generative Adversarial Network (GAN), where a generator network is trained concurrently to generate adversarial samples from noise and is updated alternately to the discriminator network. Generated samples are added to the original ones to fool the discriminator. The discriminator network is trained to distinguish the original samples from generated ones in an adversarial way. More detail is explained in section 5.3



(a) Structure of a fully connected multilayer perceptron network



(b) Structure of a Convolutional Neural Network(CNN)

Figure 2.3: A Comparison between (a) a fully connected multilayer perceptron structure and (b) a Convolutional Neural Network(CNN). Inputs in CNN are 2-D arrays (images), while inputs to the fully connected neural networks are vectors. The computations performed by both networks are very similar:(1) a sum of products is formed, (2) a bias value is added, (3) the result is passed through an activation function, and (4) the activation value becomes a single input to a following layer. The last pooled feature maps of the CNN are vectorized and serve as the input to a fully connected neural network. The class to which the input image belongs is determined by the output neuron with the highest value. Image (a) source: https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05, Image (b) source: [Gonzalez, 2009, page 965]

2.6 Convolutional Neural Network (CNN)

In MLP networks, or fully connected neural networks pattern features are designed as vectors. They were engineered and extracted from images prior to be given as input to a neural network. However, one of the main strength of neural networks is that they are capable of learning necessary pattern features directly on its own from training data. One solution to do this, is to convert images to vectors directly by organizing the pixels as the elements of the vector. However, this approach does not utilize any spatial relationships that may exist between pixels in an image[Gonzalez, 2009]. A Convolutional Neural Network (CNN or ConvNet for short) [Krizhevsky et al., 2012] is a class of deep neural networks with spatially bounded parameters, that accept images as inputs. CNNs are mainly used in Computer Vision or image processing for doing a specific task like image classification, face recognition, and semantic segmentation. The main efficiency of a CNNs is their ability to work with grid-structured inputs such as image, where there are strong spatial dependencies in local areas of the grid. For example, in a 2-dimensional image, neighbouring spatial locations in an image often have similar colour values of the individual pixels.

2.6.1 CNN Architecture Overview

Convolutional Neural Networks like almost every other neural network are comprised of multi-layer neurons with learnable weights and biases, which are trained by a kind of back-propagation algorithm [Goodfellow et al., 2016]. However, their network architecture and input differ from multi-layer perceptron networks. MLPs take a vector as input and their hidden layers are a series of the fully-connected layers, in which neurons between two adjacent layers are fully pairwise connected. The last layer called output layer is another fully connected layer that in a classification application represent the class scores (See Figure.2.3(a)).
Contrary to MLPs, input to the CNN is tensor(more than one dimension), that can understand spatial relation between nearby pixels. Figure 2.3(b) illustrate the general architecture of a CNN network. CNNs consist of one or multiple convolution layers, which extract the simple features from input by executing convolution operations. Each layer in the convolutional network is arranged according to a spatial 3-dimensional grid structure, which has a height, width, and depth. The depth of a layer in a CNN can refer to the number of channels(blue, green, and red) in the input image or the number of feature maps in the hidden layers. Moreover, the neurons in each layer are connected to a small local spatial region in the previous layer (Receptive field) instead of all neurons in a fully-connected manner. The convolution operation in each layer and the transformation to the next layer is critically dependent on maintenance of these spatial relationships among the grid cells. The final output layer present the output in form of a single vector of class scores, arranged along the depth dimension [Goodfellow et al., 2016].
The main three types of layers in CNN are *convolution*, *pooling*, and *activation* layer. Besides, the final layer is often fully connected that maps the set of inputs to a set of output nodes in an application-specific way. In the following, we will explore the

various types of layers that comprise a convolutional neural network and delve into their respective functionalities, which intertwine within the network structure.

2.6.2 CNN Layers and Operations

In Convolutional Neural Networks (CNNs), the convolutional layer plays a pivotal role by utilizing filters, also known as kernels, to conduct convolution operations on its input [Goodfellow et al., 2020][page 331]. These operations are instrumental in discerning spatial hierarchies within data, such as images, enabling the detection of specific patterns like edges or textures. The filters achieve this by systematically traversing the input, computing the dot products at each location between the filter's weights and the local regions of the input. Each filter, essentially a two-dimensional matrix of learnable weights with predefined dimensions (height $\mathrm{F_h}$ and width $\mathrm{F_w}$), adapts during the training process to optimize pattern detection.
The convolution process involves the filter moving across every possible position on the input, applying the dot product operation at each spot to generate the output feature map. Mathematically, the convolution operation conducted by a single filter at a location (i, j) on the input is represented as:

$$z_{i,j}^{[l]} = \sum_{u=0}^{F_h-1} \sum_{v=0}^{F_w-1} \boldsymbol{X}_{i+u,j+v}^{[l-1]} \cdot \boldsymbol{W}_{u,v}^{[l]} + b^{[l]} \tag{2.24}$$

In this expression, $z_{i,j}^{[l]}$ denotes the output at position (i, j) in the feature map of the l-th layer, prior to the activation function. The term $\boldsymbol{x}_{i+u,j+v}$ references the value at position $(i + u, j + v)$ in the input for layer l, marking the specific spatial region under consideration. The weight $\boldsymbol{W}_{u,v}^{[l]}$ at position (u, v) within the kernel for layer l is highlighted, with the kernel dimensions being $F_h \times F_w$. The bias term $b^{[l]}$ is added to the sum. Both $\mathbf{X_{ij}}$ and $\mathbf{W}^{[l]}$ are presented as matrices to underscore their spatial attributes, thereby signified in bold to indicate their tensorial nature rather than simple vectorial forms (see section 2.4.1).
The operation commonly referred to as "convolution" in CNN literature often resembles the mathematical operation known as cross-correlation. Traditionally, convolution involves flipping the kernel both horizontally and vertically before sliding it across the input. However, in many neural network implementations, this flipping step is omitted for efficiency, as the filters are learned during training, rendering the flipping unnecessary. This leads to a slight difference in the formula notation used to describe the operation. To align with the traditional definition of convolution, which includes flipping the kernel, the operation can also be expressed as:

$$z_{i,j}^{[l]} = \sum_{u=0}^{F_h-1} \sum_{v=0}^{F_w-1} \boldsymbol{X}_{i-u,j-v}^{[l-1]} \cdot \boldsymbol{W}_{u,v}^{[l]} + b^{[l]} \tag{2.25}$$

A convolutional layer comprises multiple such kernels, each traversing the input to compute convolutions at each step, governed by a specified stride. This sequential computation across the input surface yields a two-dimensional array of outputs for each kernel, known as a feature map, encapsulating detected feature information. This is illustrated in Figure.2.4.
Following the generation of feature maps, an element-wise non-linear activation function, such as ReLU, is applied, akin to the treatment of fully connected layer outputs. This ensemble of feature maps from various kernels enriches the network's feature detection capabilities, contributing to the nuanced understanding and processing of the input data.

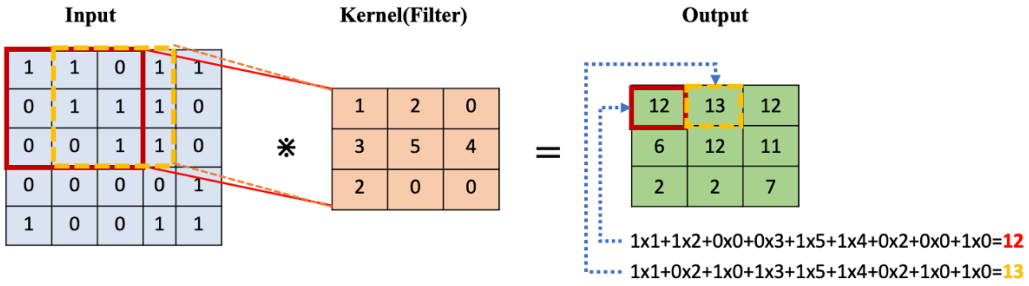


Figure 2.4: A 3×3 convolution kernel is applied across a sample image 5×5, with the step size 1, to generate an output. The resulting activation map is of the size 3×3.

Pooling layer

Like MLPs, CNNs have multiple layers that enable CNN to detect complex features. In addition to the convolutional layer, there is another type of layer called Pooling layer, which is used to decrease the spatial resolution of the feature maps. Due to the large number of weights, convolutional layers are computationally expensive and need a high spatial requirement. Pooling layers reduce the amount of the computation performed in the network by aggregating the presence of the features in a region of the feature map. common types of pooling layers are max pooling, average pooling, interpolation and stridden convolution. Irrespective of the pooling method employed, the fundamental idea behind the pooling layer is to ensure translational invariance. This is achieved by acknowledging that the *relative* spatial arrangement of features in relation to one another holds greater significance than their *precise* spatial coordinates.
Max-Pooling is a widely utilized pooling technique in CNNs. Given an input feature map $\mathbf{X}$ of dimensions $H_{\text{fm}} \times W_{\text{fm}}$ (where H_{fm} and W_{fm} represent the height and width of the feature map, respectively), max-pooling operates over a defined window of size $k \times k$ and strides across the feature map in steps of size k (assuming stride equal to the pool size for simplicity). For each $k \times k$ region, it selects the maximum

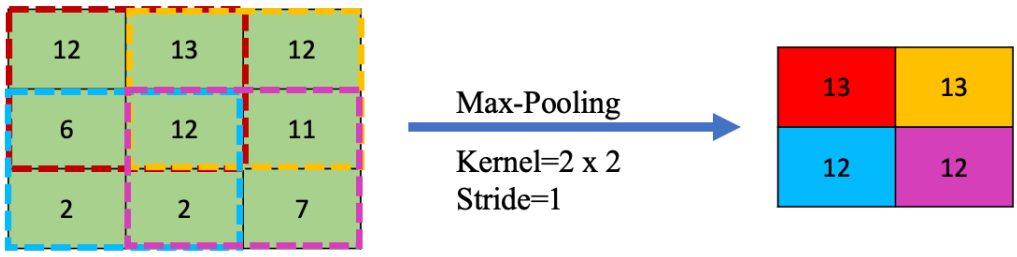


Figure 2.5: Max-pooling operation sample. The max-pooling kernel is 2×2 with the step size(stride) of 1

value within that window. The result is a new feature map with reduced dimensions $\left\lfloor \frac{H_{\text{fm}}}{k} \right\rfloor \times \left\lfloor \frac{W_{\text{fm}}}{k} \right\rfloor$, where each element in $k \times k$ block is the largest value of the region covered by the filter (See Figure.2.5).
Basically, max-pooling or average pooling are fixed operations, without having parameters need to be learnt. In [Springenberg et al., 2014] researchers raised opinion that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in the accuracy. Here, the filter kernel is shifted by a larger number than one. Replacing the convolutions with larger strides(referred to as stridden convolutions) allows the layer to learn its own pooling function instead of a fixed operation (E.g. Max pooling), which may increases the model's expressiveness ability.

Transposed Convolution Layers

Convolutional layers are well suited for feature extraction through sparse interaction and parameter sharing. If we go deeper into the network the redundancy is reduced and the features become more informative. However, By reducing the redundancy, we obtain an abstract representation of the image. For some applications like semantic segmentation, or resolution enhancement, every pixel and dimension are matters and have important information. Therefore, having same spatial dimensions of the input and output is favourable. To decompress the abstract representation and preserve the size of the input, a stack of the transposed convolution layers, which are also known as deconvolution layers, are used in CNN to reverse the spatial transformation effect of a convolution layer [Goodfellow et al., 2016]. They trained jointly with convolutional layers during the training process to learn a set of weights to reconstruct the original inputs. Transposed convolutions expand the spatial dimensions of their input, typically a feature map. This is in contrast to standard convolution operations, that tend to reduce the input dimensions. The enlargement effect of transposed convolutions makes them particularly useful in tasks such as image segmentation and generative models, where reconstructing or upsampling to a larger spatial resolution is

required. The spatial dimensions of the output produced by a transposed convolution layer can be determined using the formula:

$$\begin{aligned} H_{\text{out}} &= (H_{\text{fm}} - 1) \times s + F_{\text{h}} \\ W_{\text{out}} &= (W_{\text{fm}} - 1) \times s + F_{\text{w}} \end{aligned} \tag{2.26}$$

Here, H_{out} and W_{out} represent the height and width of the output feature map, respectively. H_{fm} and W_{fm} denote the height and width of the input feature map. s stands for the stride of the transposed convolution, and F_{h} and F_{w} are the height and width of the kernel (filter) used in the operation. By employing transposed convolution layers, the abstract representation can be decompressed, and the original input size can be preserved. Figure 2.6 visually illustrates the process of this reverse transformation.

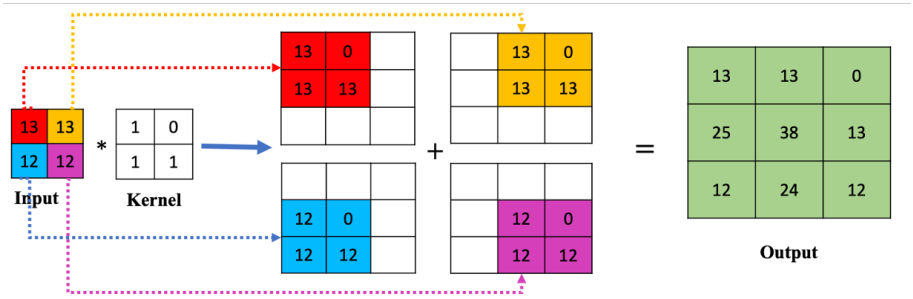


Figure 2.6: Transposed convolution operation. In this illustration, a 2×2 feature map is subjected to a transposed convolution operation using a 2×2 kernel. The transposed convolution involves matrix multiplications by aligning the kernel with each 2×2 patch of the input feature map and computing the element-wise product, followed by the summation of these products to obtain each element of the output. This process is repeated across the input, resulting in a larger output feature map with dimensions 3×3. The transposed convolution operation effectively expands the feature map, preserving spatial information and allowing for more detailed analysis in applications such as semantic segmentation and resolution enhancement.

Fully Connected Layers

The fully connected layer or Dense Layer is a layer, in which every node in the layer is connected to every node in the previous layer [Aggarwal et al., 2018]. This layer acts exactly the same as a traditional feed-forward network. Generally, more than one fully connected layer can be used in the network to increase the power of the computations towards the end. Because each node in this level receives input from all nodes in the previous level, each node can potentially have a full perspective of all features computed in the previous level. For that reason, in the image classification

applications in CNNs, this layer is mostly used as the last layer to make decisions related to the classification task.

2.6.3 Convolutional Neural Network Training

Training a Convolutional Neural Network (CNN) centers on optimizing the network's weights and biases to reduce the loss function, highlighting discrepancies between predictions and actual targets. This begins with initializing parameters, potentially leveraging specific strategies to enhance the learning phase. A CNN operates in two phases, feature extraction through convolutional and pooling layers, and classification via fully connected layers that interpret these features to produce output scores. In each training step, an input (like an image) undergoes a series of transformations through convolutional, activation, pooling, and fully connected layers, resulting in a prediction. The difference between this prediction and the actual target is quantified using a loss function, such as Cross-Entropy for classification or Mean Squared Error for regression tasks.
The training's backbone is Gradient Descent (GD), also known as the batch gradient descent [Goodfellow et al., 2016, pages 82 and 151]. It is an optimization technique aimed at minimizing the cost function by iteratively adjusting network parameters based on the entire dataset. This involves computing gradients through back-propagation and updating parameters to reduce loss, a process repeated across numerous epochs (For further details on the back-propagation algorithm, please refer to section 2.4.2).
To avert over-fitting, regularization techniques like L1/L2 or dropout may be incorporated, adding penalties to the cost function or intermittently excluding units during training to promote the learning of generalized features. The training persists until the loss function stabilizes or a predetermined epoch count is reached. Subsequently, the network's performance is evaluated on a separate dataset to confirm its generalization skills. Throughout, adjusting hyper parameters and the network's structure is key to achieving the best performance.

Training a Convolutional Neural Network (CNN) efficiently, especially with large datasets, poses challenges with computational cost and practicality. Traditional Gradient Descent (GD), which updates parameters using the entire dataset in each iteration, becomes less feasible due to the extensive computational resources required. Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951] offers a solution by updating parameters for each training example individually, thus significantly reducing the computational load per iteration. Unlike GD, which computes the cost function across the entire dataset, SGD evaluates the cost for a single, randomly selected training example at each step. This approach not only accelerates the training process but also introduces randomness that can help escape local minima. However, the variance in updates due to SGD's stochastic nature can lead to fluctuating convergence paths. To mitigate this and balance the efficiency of SGD with the sta-

bility of GD, Mini-Batch Stochastic Gradient Descent (Mini-Batch SGD) is widely adopted. This variant processes a small, random subset of the training data in each iteration, offering a compromise between the exhaustive computation of GD and the high variance of SGD. Mini-Batch SGD strikes a balance by reducing the variance of parameter updates compared to SGD, while being more computationally efficient than GD. This method is particularly favoured in CNN training for its effectiveness in handling large datasets and facilitating faster convergence.

2.7 Semantic Segmentation Background

Image segmentation is one of the fundamental step of the digital image processing. Image segmentation is a method, where a digital image is divided into multiple regions(sets of pixels) called image segments to analyse them with respect to a particular application such as classification, pattern recognition [Rafael, 1992]. It is one of the long standing Computer Vision problem. Different methods have been proposed over decades to improve the segmentation results, starting from classical methods such as threshold-based techniques, to the graphical-based models and finally deep neural networks, which are performing a semantic segmentation with understanding of image in pixel level with outstanding results. In this section, we review the fundamental and well-known image segmentation techniques. We briefly discuss taxonomic of image segmentation techniques. Then we focus on the semantic segmentation techniques based on the deep learning and discuss in detail two state of the arts deep learning methods, which are widely used for semantic segmentation task.

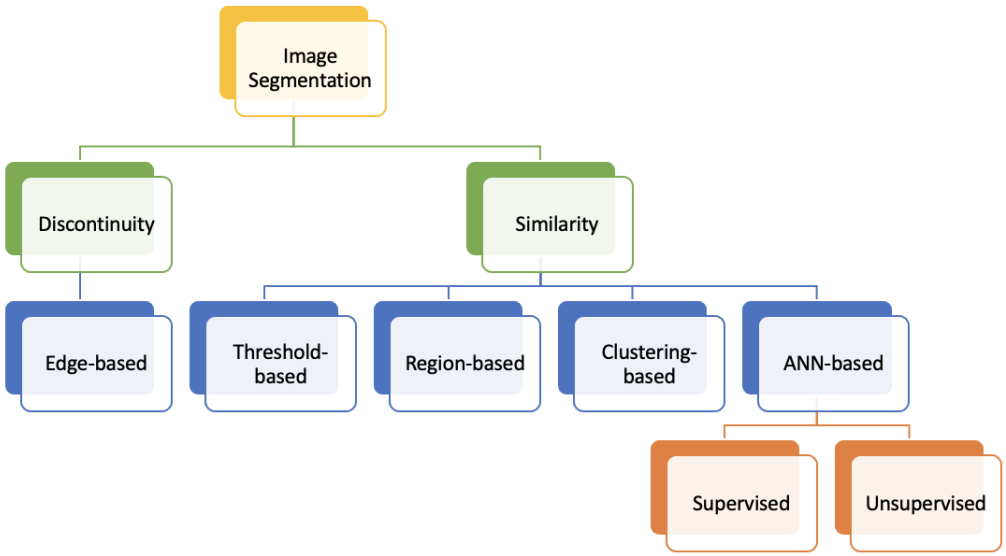


Figure 2.7: Image segmentation categories,adapted from [Saeed, 2020]

2.7.1 Taxonomy of image segmentation methods

Fu et al. [Fu and Mui, 1981] discuss a survey on different image segmentation methods before the deep learning era. Figure 2.7 displays the taxonomy of those techniques. Generally, image segmentation methods are divide into 6 groups based on their characteristics

1. Threshold-based Segmentation
2. Edge-based Segmentation
3. Region-based Segmentation
4. Clustering-based Segmentation
5. Graph-based Segmentation
6. ANN-based Segmentation

Threshold-based Segmentation

It is a global technique of image segmentation, where separating foreground or object from the background with no overlapping sections [Sahoo et al., 1988]. This requires a certain amount of the contrast between the subject and the background. In this technique, the intensity histogram of all the pixels in the image is considered. Then a threshold value is set to divide the image into sections. However, this threshold has to be perfectly chosen to segment an image into an object and a background. Various threshold techniques are Global, local and Split, merge and growing [Zaitoun and Aqel, 2015].

Edge-based Segmentation

Another segmentation technique is edge detection, which relies on finding edges in an image by using different edge detection techniques [Zaitoun and Aqel, 2015]. Edge detection is sensitive to relative changes at a local level i.e the boundaries of objects and regions. By moving from one region to another (or from the subject to the background), the gray level may change and a shift in the gray levels can be detected. Although edge detection methods perform better than threshold-based techniques, they can only describe the strength and direction of edges of neighbouring pixels, where various subsequent information are required to link the edge fragments and to extract the desired features like lines, open curves, shape boundaries, etc [Bali and Singh, 2015]. A variety of different edge detection operators with different mathematical and computational costs have been developed such as Sobel [Kanopoulos et al., 1988], Prewitt [Prewitt et al., 1970], Canny [Li et al., 2009],etc.

Region-based Segmentation

It divides the entire image into sub regions or clusters with the same grey level for all pixels in one region. In this method, neighbouring pixels are scanned and grouped to a

region based on a given homogeneity criteria. In other word, the similarity distance of each pixel should satisfy the homogeneity criteria. Watershed segmentation [Beucher and Lantuejoul, 1979], Split-and-merge [Pavlidis and Horowitz, 1974] and region-growing techniques [Ikonomatakis et al., 1997] are examples of such methods.

Clustering-based Segmentation

Clustering algorithms are used to group similar pixels based on similar features, such that pixels in a similar group(cluster) are more comparative to each other rather than those in dissimilar groups. Features are parts of information that can be the extracted from an image processing or computational task based on attributes of image properties. There are a large variety of features used for image segmentation and other purposes, such as Pixel colour, Histogram of oriented gradients (HOG) [Dalal and Triggs, 2005], Local Binary Pattern (LBP) [Ojala et al., 1994]. In clustering approaches, one or more particular features can be extracted and then group them based on a specific distance metric system to aggregate the feature samples into homogeneous regions. Hierarchical clustering, K-Means, Fuzzy C-Means are some examples of the clustering-based methods [Dubey et al., 2018].

Graph-based Segmentation

It comprises a group of image segmentation algorithms based on graph theory, where an image is considered as a graph. The vertices of the graph represent each pixel of the image and edges link adjacent pixels. Weights on the edges are assigned in accordance with the similarity between two neighbouring pixels like colour, distance, or textures. Comparison to the previous categories of image segmentation methods, graph-based methods can capture precise definitions of the properties of a group, which often reflect global aspects of the image and result in a better segmentation. In addition, according to their graph structure, all image features such as intensity, colour texture, continuity of edges are treated in a uniform network. While research over the past years have seen considerable progress in graph-based methods of image segmentation, these methods have still complex computation and are too slow to be practical for many applications. These graph based segmentation methods might be grouped as a)graph cut based methods, b)interactive methods, c) minimum spanning tree based methods and 4) pyramid based methods [Camilus and Govindan, 2012].

ANN-based Segmentation

Artificial Neural Networks (ANN) have emerged as a highly sought-after method for image segmentation, particularly in the realms of semantic and instance segmentation, leveraging the power of deep neural networks. Their outstanding features, such as the ability to gracefully handle noise, real-time applicability, and exceptional generalizability, have fuelled an unprecedented boom in the adoption of ANN-based methods for achieving precise and accurate image segmentation. The ANN-based image segmentation techniques are mainly divided into two categories [Amza, 2012]: supervised and unsupervised methods. In supervised approach the training data and available

ground truth are used to train the machine. Unsupervised methods or clustering processes are semi or fully automatic in the absence of the ground truth. However, these architectures may be implemented using prior application-specific knowledge at design time, i.e., anatomical, physical, or biological knowledge.
In the next section, we focus on semantic segmentation based on the deep learning techniques and discuss two main approaches which are widely used as the bases in most semantic segmentation applications.

2.7.2 Semantic Segmentation Using CNNs

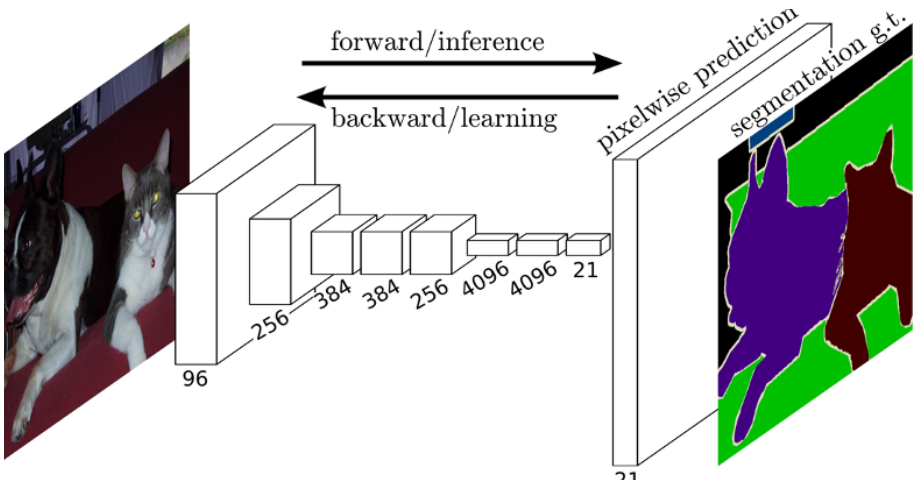
As mentioned before, semantic segmentation provides a label at the pixel level. The improvements of semantic segmentation techniques facilitate the solution of real-word applications such as medical imaging [Ronneberger et al., 2015], autonomous driving [Badrinarayanan et al., 2017] or even paving the way to new ones. Like other scene understanding tasks, semantic segmentation have moved from classical method using hand-crafted features [Fu and Mui, 1981] to deep learning based techniques, which obtain impressive state-of-the-art results [Garcia-Garcia et al., 2017]. In the last few years, semantic segmentation has attracted considerable attention and made great progress in many vision tasks [Krizhevsky et al., 2012, P. Sermanet, 2013, Long et al., 2015], especially due to the performance enhancement of CNNs. In [Zhu et al., 2016] is discussed a in-depth review of recent development of image segmentation methods, including classic bottom-up methods, interactive methods, object region proposals, semantic parsing and image co-segmentation, which provides a detailed comparison of classical solutions for semantic segmentation with special emphasis on superpixel methods. In [Garcia-Garcia et al., 2017], comprehensive advancements in deep learning-based approaches for semantic segmentation are presented, encompassing novel architectures and commonly utilized datasets. The forefront of semantic image segmentation is dominated by deep learning techniques, where the evolution of architectures has transitioned from CNN-based structures primarily designed for classification tasks to adaptive architectures tailored specifically for segmentation. These evolved architectures enhance the resolution of learned feature representations by incorporating a decoder module atop the classification CNN, enabling pixel-level predictions. The encoder, or base classifier architecture, learns features while progressively reducing resolution. In these evolved architectures, an additional decoder module further enhances the resolution of the learned feature representations, facilitating per-pixel predictions. Noteworthy among the various CNN-based architectures proposed for semantic segmentation are the influential solutions of Fully Convolutional Neural Network (FCN) [Long et al., 2015] and U-Net [Ronneberger et al., 2015], which have paved the way for modern semantic segmentation architectures.
Long et al. [Long et al., 2015] introduced a groundbreaking technique called fully convolutional networks (FCN) for semantic segmentation. Unlike patch-wise training methods, FCN utilizes the entire image to derive dense predictions. It achieves this by transforming the fully connected layers of the CNN classifier into convolutional layers with appropriate filter sizes. To upsample the learned feature map from the CNN classifier to the input resolution, FCN employs bi-linear interpolation. Fur-

thermore, the method utilizes transposed convolutions to upsample feature maps at multiple scales, which are then merged into the final prediction. For per-pixel labelling, a standard classification loss, such as the Cross-Entropy (CE) loss [De Boer et al., 2005], is applied independently to each pixel. This loss function operates on individual pixels and does not take neighbouring pixel information into account.
Noh et al. [Noh et al., 2015] introduced the Deconvolutional Network (DeconvNet) for semantic segmentation, featuring an advanced decoder network. This network architecture utilizes a series of deconvolution (or un-pooling) and Rectified Linear Unit (ReLU) layers in a stacked configuration. The design facilitates gradual upsampling, interspersed with intermediate convolutions, along the expanding path of the network. This approach marks a significant development in the field of semantic segmentation. SegNet [Badrinarayanan et al., 2017] is another alternative architecture for semantic segmentation, in which a symmetric encoder-decoder architecture is defined. Contrary to DeconvNet, during upsampling, the max pooling indices at the corresponding encoder layer are recalled to upsample.
In U-Net [Ronneberger et al., 2015], a u-shaped architecture network was proposed with long-range skip connections between the contracting and expanding path. The feature maps from different encoding layers are concatenated with the upsampled feature maps from the corresponding decoding layers. Therefore U-Net is able to preserve fine detail, while, incorporating context from a larger receptive field. Since in U-Net, the entire feature maps are transferred from the encoder path to the decoder path, a lot of memory is used, while in SegNet only the pooling indices are transferred to the expansion path from the compression path using less memory.
In addition to the architectures previously mentioned, Residual Networks (ResNets) stand out as a key development in the realm of semantic segmentation. ResNets [He et al., 2016] brought a novel approach to network design through the implementation of residual blocks. These blocks are designed to combat the vanishing gradient problem, a common issue in deep networks, by introducing '*shortcut connections*'. In a ResNet, each layer's output is fed not only into the next layer but also 'skips' ahead to layers approximately 2–3 hops away. A hop is a single layer or a set of operations in the network. This innovative design, where the output of one layer is connected to another layer several hops ahead, facilitates the training of much deeper networks. It does so by allowing the flow of gradients through these shortcuts, ensuring effective learning even in very deep layers.
Building on the foundational architectures of FCN, DeconvNet, SegNet, and particularly U-Net and ResNets, the field of semantic segmentation has experienced a significant evolution towards optimizing network architectures, each with their unique impact on efficiency and performance. ResNets, known for their *shortcut connections*, allow the construction of deeper networks to capture more complex features. However, this increased depth can lead to higher computational complexity compared to architectures like U-Net. While U-Net, with its U-shaped architecture and long-range skip connections, excels in preserving fine detail and incorporating a larger receptive field context, it manages to do so with a comparatively moderate computational demand. This is in contrast to the potentially higher computational requirements of deep ResNets. Nonetheless, the innovation of ResNets lies in their ability to ad-

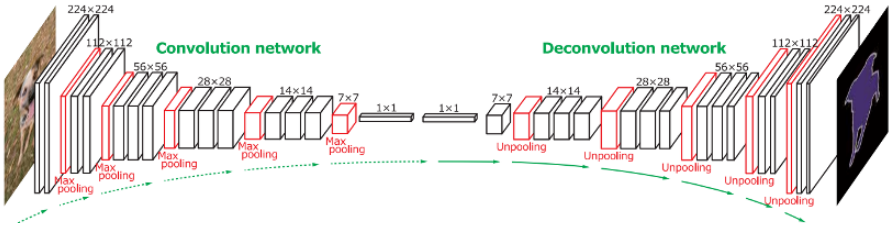
dress the vanishing gradient problem, enabling the training of deeper models that were previously not feasible. SegNet distinguishes itself with its memory-efficient design, which hinges on transferring only pooling indices, unlike U-Net. This approach starkly contrasts with U-Net's method of transferring entire feature maps from the encoder to the decoder, a process that significantly increases memory usage. As illustrated in figures 2.8 and 2.9, these developments show the evolution of semantic segmentation architectures, each building upon and refining the concepts of its predecessors. Despite the high performance of these models, their reliance on extensive, fully supervised annotated data presents significant challenges. These challenges are particularly acute in scenarios, where such data is scarce or the high costs of semantic segmentation annotations are prohibitive.
In response to these challenges, newer architectures like ENet [Paszke et al., 2016], ERFNet [Romera et al., 2017], and ICNet [Zhao et al., 2018] have emerged, prioritizing processing speed and efficiency. These models achieve this by operating at lower resolutions, facilitated by rapid downsampling at the initial stages of the encoder and employing lightweight decoders to reduce network parameters. This approach marks a significant departure from the more memory-intensive methods like U-Net, aiming to balance the trade-off between efficiency and accuracy. While these advancements substantially enhance efficiency, they often come with a compromise in accuracy. Moreover, the pursuit of real-time performance in per-pixel labelling methods remains a challenge due to the high computational requirements, particularly in embedded systems and advanced driver assistance systems (ADAS). The upcoming chapter delves deeper into these challenges faced by semantic segmentation in real-time applications and discusses the contributions of this thesis in addressing these complexities.

2.8 Summary

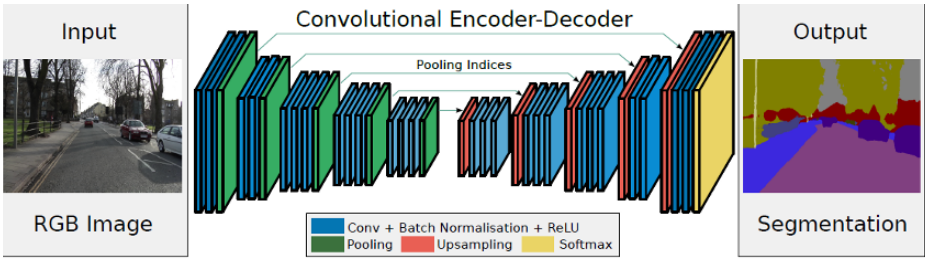
In this chapter, we have covered the technical backgrounds that have been used in the following chapters. In particular, we explained the fundamental definitions and concepts about the artificial Neural networks, with focus on deep neural networks and investigated the general strategies for training of machine learning models. We presented the major building blocks of a convolutional neural network architecture, which has been widely used as the learning approach in this thesis. We also reviewed the main concepts and the state-of-the arts methods in the tasks related to the semantic image segmentation. In fact, it was a broad area, and it was impossible to review all related methods. We illustrated the taxonomy of the different approaches in the image segmentation applications, started with the classical methods to the most recently ones based on deep learning. We will also review the most related methods to our contributions in the later chapters, where it is necessary.



(a) Fully Convolutional Neural Network(FCN) Architecture [Long et al., 2015]

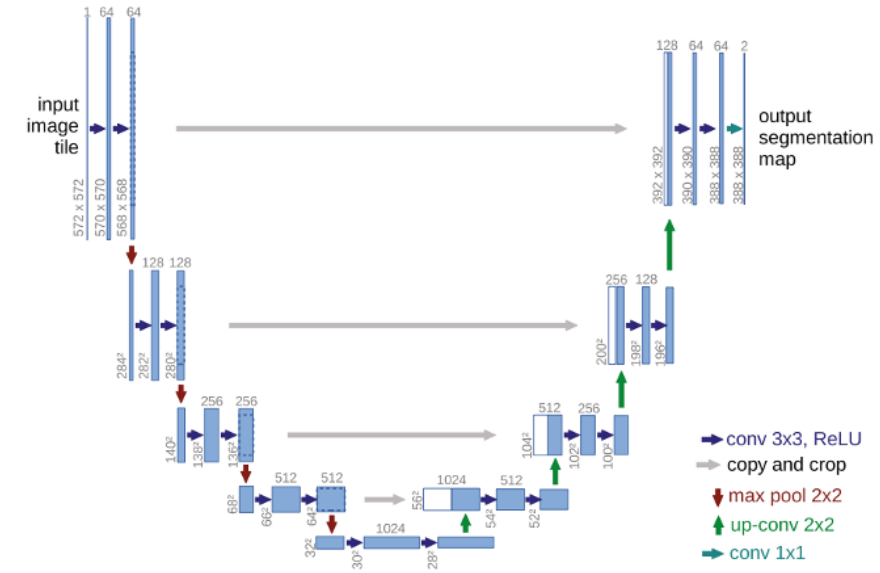


(b) DeconvNet Architecture [Noh et al., 2015]

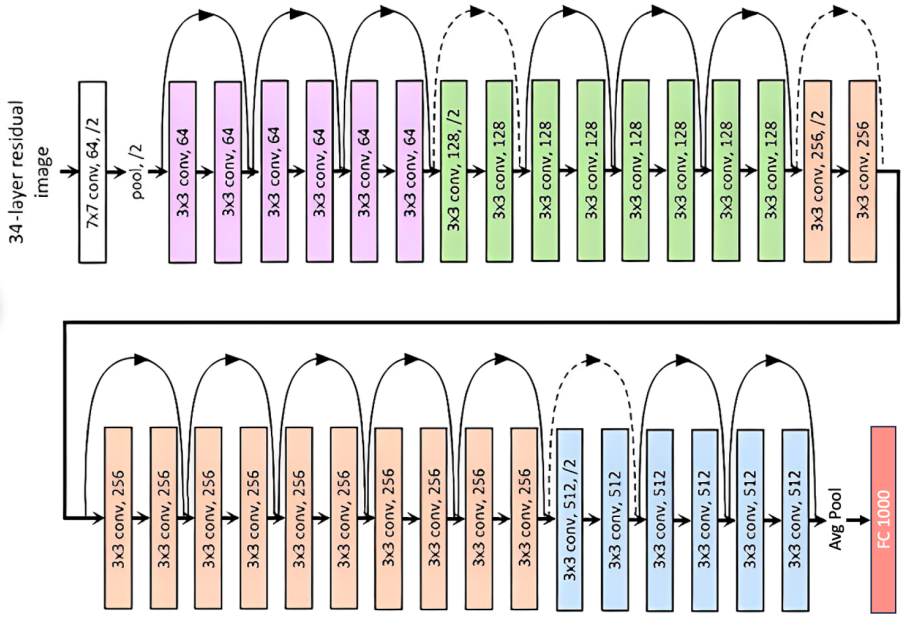


(c) SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation [Badrinarayanan et al., 2017]

Figure 2.8: Representative Examples of State-of-the-Art Semantic Segmentation Architectures (Part 1): FCN, DeconvNet and SegNet.



(a) U-Net Architecture [Ronneberger et al., 2015]



(b) ResNet Architecture [He et al., 2016, Bangar, 2022]

Figure 2.9: Representative Examples of State-of-the-Art Semantic Segmentation Architectures (Part 2): U-Net, and ResNet.

3 Superpixel-based Road Segmentation for Real-time systems using CNN

Convolutional Neural Networks (CNN) contributed considerable improvements for image segmentation tasks in the field of Computer Vision. Despite of their success, an inherent challenge is the trade-off between accuracy and computational cost. The high computational efforts for large networks operating on the image's pixel grid makes them ineligible for many real time applications such as various Advanced Driver Assistance Systems (ADAS). The purpose of this chapter is to present and discuss the first contribution of this thesis. As mentioned in the introductory chapter, our first contribution focuses on the learning a supervised semantic segmentation model with a time and resource budget, which makes it suitable to be used successfully for real-time applications. In particular, this contribution has been developed for the use case of autonomous driving applications like road segmentation. Because in this area, a precise pixel-by-pixel classification of the image with more attention to the computationally efficient solutions is crucial.
In this chapter, we propose a novel supervised CNN approach for semantic segmentation task, based on the combination of superpixels and high dimensional feature channels. We provide an example of applying the proposed method for semantic segmentation of urban scene images, targeting the application of driver assistance systems.

In the upcoming sections of this chapter, we will delve into a detailed discussion of the proposed method, focusing on its application to two open-source imagery datasets within the context of autonomous driving. The core idea is to reduce the computational complexity by segmenting the image into homogeneous regions (superpixels) and feed image descriptors extracted from these regions into a CNN rather than working on the pixel grid directly. To enable the necessary convolution operations on the irregular superpixels, we introduce a lattice projection scheme as part of the superpixel creation method, which composes neighbourhood relations and forces the topology to stay fixed during the segmentation process. Reducing the input to the superpixel domain allows the CNN's structure to stay small and efficient to compute, while keeping the advantage of convolutional layers. The method is generic and can be easily generalized for segmentation tasks other than road segmentation.

The works of this chapter are partly published in [Zohourian et al., 2018b] and [Zohourian et al., 2018a].

3.1 Introduction

One of the long-lasting goals of Computer Vision is the automated scene understanding from a variety of the images. Exposing image specification is useful for applications, like image editing, image search, and environment perception for autonomous vehicles. Detecting objects like roads, pedestrians, vehicles, traffic signs, etc. is important for many driver-less cars and driver assistance systems. Due to the variability of different factors like colour, shape, illumination and shadows or obstacles on the road surface, the road detection is a challenging problem. The state-of-the-arts techniques to solve this problem are mainly based on deep learning and Convolutional Neural Networks (CNNs) [LeCun et al., 2015, Schmidhuber, 2015]. These methods enable towards better visual understanding by applying a semantic segmentation process in which each pixel is assigned to an object category. The segmentation result provides meaningful information to support higher level scene understanding tasks.
As we discussed in 2.7.2, there are two major approaches to train CNN-based image processing systems. The two approaches differ with respect to the input data model. One of the approaches is based on a patch-wise analysis of the images, i.e. an extraction and classification of rectangular regions having a fixed size for every single image [Ciresan et al., 2012, Farabet et al., 2013, Ganin and Lempitsky, 2014, Ning et al., 2005]. The other one is based on full image resolution, wherein all pixels of an image in the original size are analysed [Long et al., 2015]. Most recent improvements in both CNN-based methods were accomplished by increasing the network size [Simonyan and Zisserman, 2014, He et al., 2016], whereas deeper networks provoke large computational costs that make them unsuitable for embedded devices in driver assistance systems.
In the current work we apply a superpixel-based CNN method for the specific application of pixel-wise road segmentation that uses superpixels as input data model. To the best of our knowledge, it is the first time that irregular superpixels with regular lattice projection for convolutional purpose is given as input data model into a CNN network. The proposed method comprises the following steps:

- First, segmenting the image into superpixels, wherein the superpixels are coherent image regions comprising a plurality of pixels having similar image features.
- Then determining image descriptors for the superpixels, wherein each image descriptor comprises a plurality of image features.
- The superpixels are assigned to the corresponding positions of a regular grid structure extending across the image.
- This lattice together with the image descriptors are fed to the convolutional neural network, to classify the superpixels of the image according to semantic categories.

Feeding a network with almost well segmented " superpixel" units enables the network to learn local information like contrast, shape, texture, etc. much better rather than

using raw image pixels. In comparison to [Long et al., 2015], that is based on full resolution input data and has a deep convolutional network layering (e.g: vgg-19), our method combines larger basic units " superpixels " with simple network structure. This results in significant reduction of the computational costs for a densely labelled map prediction. Contrary to patch-based semantic segmentation approaches [Farabet et al., 2013], information about spatial context in the proposed method can be preserved preferably due to the usage of superpixels.

The subsequent sections of this chapter are structured as follows:

In Section 3.2, we provide an overview of relevant literature and discuss prior works in the field. Section 3.3 focuses on our novel approach, the superpixel-based Convolutional Neural Network (SP-CNN). Here, we delve into the details of the superpixel segmentation method and elaborate on the proposed high-dimensional feature descriptor. In Section 3.4, we begin by describing the two primary publicly available datasets used in our study, which are specifically designed for self-driving applications. Subsequently, we analyse the proposed model using two distinct machine learning approaches. Furthermore, we delve into the architectural aspects of the proposed network and discuss the selection of relevant parameters. Section 3.5 presents the experimental results obtained from the aforementioned datasets, focusing on urban scene images captured from various visual perspectives. The evaluation process encompasses accuracy assessment as well as computational time analysis. Finally, Section 3.6 draws conclusions based on the findings and discuss potential future works and areas that require further improvement.

3.2 Related Works

Road segmentation has been studied for decades. It is essential in autonomous driving and mobile robot applications. Traditional road segmentation techniques mostly rely on the detection and extraction of local characteristics in images such as colour, position prior, edge, texture [Alvarez and Ĺopez, 2010, Sturgess et al., 2009, Lombardi et al., 2005]. Alvarez et al.[Alvarez and Ĺopez, 2010] separate road from background by using illumination-invariant models. Sturgess et al. [Sturgess et al., 2009] proposed a novel framework to combine motion and appearance features for object class segmentation problems. Their approach classify the road based on the motion and appearance features and then refine the road boundary with the conditional random field. A road model switching is proposed in [Lombardi et al., 2005] to fit the road model to the road configuration. Numerous hand-crafted low-level features have been suggested in previous studies to address particular challenges such as occlusions, scale variations, and illumination differences. Nevertheless, these features often fail to handle illumination changes or effectively handle occlusions caused by objects like vehicles or pedestrians on the road surface. Consequently, the traditional methods exhibit limited generalizability, leading to imprecise outcomes in road segmentation. Recently, deep learning has made remarkable progress in Computer Vision. Developing CNNs [LeCun et al., 2015] had the biggest impact on this success for tasks

such as image classification [Krizhevsky et al., 2012], object detection [Girshick et al., 2014, P. Sermanet, 2013], scene labelling [Farabet et al., 2013, Chen et al., 2014]. Semantic segmentation research has experienced a huge improvement in the era of deep learning. The state of- the-art methods for semantic segmentation are generally fully convolutional networks [Long et al., 2015], which are directly applied to the whole image. This method has been improved further in several newer approaches such as " DeepLab " [Chen et al., 2014]or SegNet [Badrinarayanan et al., 2015]. However, some aspects for semantic segmentation such as computational efficiency has not been thoroughly investigated in the literature. Although, this would have a huge impact on applications such as autonomous driving. Most of the per pixel labelling methods are too expensive for embedded applications and they require powerful GPUs to be fast enough for achieving the real-time performance.
In this work we combine superpixels segmentation with convolutional neural network. Several other methods benefit from this combination too. Gadde [Gadde et al., 2016] embedded superpixels into a newly defined layer that he names "Bilateral Inception" which acts as an edge preserving filter. This layer is substituted with a fully connected layer and propagates label information between superpixels. This results in better segmentation than in exclusively pixel-wise approaches. However, this network still uses full resolution images as the inputs. SuperCNN [He et al., 2015b] is a neural network based approach for salient object detection. A sequence of superpixels, instead of a 2D image pattern, is fed into this network. Contrary to this 1-D inputs, our proposed method uses a 2D-grid of superpixels as input to the network which allows for easier extraction of the neighbourhood information by convolutional network.

3.3 Proposed Method: Real-time Superpixel-CNN Semantic Segmentation Framework

This work addresses the task of road segmentation from urban scene images. We tackle the problem by segmenting the images into superpixels, deriving road relevant features, and constructing a rational feature model fed into CNN to segment road regions. Figure 3.1 displays the architecture of our method. superpixels are extracted using the Simple Linear Iterative Clustering (SLIC) [Achanta et al.,] algorithm (see section 3.3.1). The main features extracted from the superpixels are intensity, texture and location (see section 3.3.2). The proposed idea is applied into two different machine learning methods including support vector machine (SVM)[Cortes and Vapnik, 1995] and convolutional neural network [Goodfellow et al., 2016], which are explained in section 3.4.2.

Figure 3.1: The Architecture of the Proposed superpixel-based convolutional neural network for semantic segmentation task. The process begins with the segmentation of the original image into inhomogeneous superpixels. Local descriptors are then extracted for each superpixel. These descriptors are assigned to their corresponding nodes in the lattice projection. Together, the lattice projection and the superpixel descriptors are fed into a CNN for the classification of superpixels according to the semantic categories. In the lattice projection, the red nodes represent the centres of the inhomogeneous superpixels, serving as their representatives. The blue lines connect adjacent nodes, forming a connectivity structure within the lattice projection.

3.3.1 superpixel Extraction

In this part we discuss the superpixels concept as the basic units in our proposed method. We explain which method we used for extracting the superpixels and how we adapted them for embedding in convolutional neural network.
Superpixel segmentation methods are employed to partition an image into spatially homogeneous regions, which are then treated as cohesive entities for subsequent processing in image processing and Computer Vision applications (see section 2.7.1). Compared to individual pixels, superpixels hold more perceptual meaning and offer significant computational efficiency gains [Ren and Malik, 2003]. While superpixel segmentation techniques aim to achieve homogeneous areas, challenges arise due to factors such as scene characteristics, spectral variability, and the tuning of scale parameters in the extraction algorithm. To address these challenges, superpixel extraction methods incorporate spatial connectivity and spectral similarity, effectively grouping neighbouring pixels with similar characteristics. Evaluating the performance of superpixel segmentation can be done based on criteria such as runtime, error metrics, and segmentation robustness. Adherence to the object boundary, simplicity, memory efficiency and speed are several criteria, that can affect the performance of the superpixel segmentation method [Ren and Malik, 2003]. Therefore, the selection and tuning of the superpixel algorithm and its parameters are crucial for achieving optimal results in specific applications. Two primary advantages of well-extracted superpixel properties, which motivate us to adopt them as fundamental units in our approach, are outlined below:

- **Accuracy** Well segmented superpixels can store more compact information about the color, texture, etc. and they are less ambiguous and sensitive to noise than features extraction at pixel level. They can preserve the geometric object structures and adjust well to the object contours.
- **Efficiency** Dealing with millions of pixels and their parameters(such as spatial coordinates, intensity levels, etc.) in large systems can be costly, whereas using superpixels can greatly reduce the model complexity and computation cost especially for real time systems.

Regular superpixel segmentation methods, characterized by a grid-based arrangement of superpixels, are commonly employed to preserve the topology of an image. However, this regularity comes with several drawbacks. Firstly, it can compromise the maximum homogeneity of texture within each superpixel, as the grid structure may not align well with the actual texture boundaries. This limitation leads to mixed or inconsistent representations of texture within the superpixels. Secondly, regular superpixel segmentation often oversimplifies complex object boundaries, resulting in a loss of detail and inaccurate delineation. Moreover, the fixed grid structure may not be the most efficient representation for images with irregular or non-uniform structures, limiting its applicability. Additionally, regular superpixel segmentation lacks the flexibility to adapt to diverse image characteristics and different levels of

detail. Objects with varying scales, shapes, or orientations may not be adequately captured by the fixed grid structure, further compromising the segmentation quality. Lastly, regular superpixel segmentation methods heavily rely on parameter tuning, such as the size of the grid cells, to achieve optimal results. However, finding the right parameter values that suit a wide range of images can be challenging, and suboptimal choices may lead to inadequate segmentation quality. Considering these drawbacks, irregular superpixel segmentation methods are often preferred as they offer more flexibility in capturing complex structures, preserving texture homogeneity, accommodating diverse image characteristics, and reducing sensitivity to parameter tuning. However, superpixels generated through irregular segmentation have varying sizes and irregularly shaped boundaries, making them unsuitable as direct input for convolutional networks. To utilize convolutional operations and kernels efficiently, a regular topology is required. Therefore, a necessary step involves re-aligning or adjusting the irregular superpixels to conform to a regular structure, enabling them to serve as suitable input for a convolutional network.

Original SLIC Method

There are different superpixel algorithms. Region-based [Yu, 2005], mean shift [Comaniciu and Meer, 2002], water shed [Vincent and Soille, 1991] and graph-based [Felzenszwalb and Huttenlocher, 2004] approaches are well-known techniques for superpixel extraction with the high performance. They have their own pros and cons. We used Simple Linear Iterative Clustering (SLIC) algorithm [Achanta et al.,] for superpixel segmentation for all our experiments. It has been shown to provide competitive results with a minimum of computational complexity among other superpixel extraction methods.
Figure 3.2 shows an example of applying SLIC on one of our image. Algorithmically, SLIC simply computes a k-means clustering over pixels. SLIC initiates with equally-sized superpixels arranged in a grid structure. The similarity between pixels is calculated based on two criteria: spectral similarity and spatial proximity that enforces compactness and regularity in the superpixel shapes. The main idea of this approach is to limit the search space to a region proportional to the desired SP size which reduces considerably the calculation time. superpixels grow by measuring the (spectral-spatial) distance between each pixel to its cluster centre and then update the cluster centres based on K-means algorithm.The input parameters for this method include the input images with a total of N_{px} pixels, the desired number of approximately equally-sized superpixels represented by N_{sp}, and a weighted distance denoted as m_{slic}. The value of m_{slic} serves as a control factor to adjust the compactness of the superpixels by balancing the trade-off between colour similarity and spatial proximity in the distance calculation. A higher value of m_{slic} places greater importance on spatial proximity, leading to the creation of more compact superpixels. Conversely, a lower value of m_{slic} emphasizes colour similarity over spatial proximity.

To begin, the superpixels are initialized with approximately equal sizes, each containing approximately N_{px}/N_{sp} pixels. When aiming for roughly equally sized super-

pixels, a superpixel center is positioned at regular interval $\mathrm{S} = \sqrt{N_{px}/N_{sp}}$ on a grid. Here, S represents the interval distance between the centers of adjacent superpixels on a grid. Each superpixel has a spatial extent of approximately S^2, representing its neighbourhood area. The SLIC algorithm specifically calculates distances from each cluster center to the pixels located within a $2S \times 2S$ area. This ensures that the pixels associated with a particular cluster center reside within this defined area and are not situated farther away. It leads to reduction of complexity and distance computations, independent from the number of superpixels [Achanta et al.,].
For a set of superpixel centers, each represented by $\mathrm{Cent_i} = [l_i, a_i, b_i, x_{p_i}, y_{p_i}]$, where i ranges from 1 to N_{sp} and each positioned at consistent intervals S on the grid, the algorithm determines the closeness of pixels to their respective cluster centers. This is achieved by measuring Euclidean distances in two realms, the CIELAB color space, where l_i, a_i, b_i denote the lightness and color-opponent dimensions to mimic human visual perception, and the spatial domain, where $\boldsymbol{P} = (\mathrm{x_p}, \mathrm{y_p})_i$ pinpoint the i-th superpixel center's location on the image plane. Such a dual approach ensures a balanced consideration of color similarity and spatial proximity. However, to avoid the potential for uneven superpixel shapes that can emerge from diverse superpixel sizes, the algorithm incorporates a compactness control $\mathrm{m_{slic}}$ as outlined in the equation 3.1. Larger m_{slic} resulting superpixels are more compact and smaller m_{slic} aimed better segmentation, but more irregular size and shape. The authors in [Achanta et al.,] mentioned, that the value of m_{slic} can be chosen from the range of $[1, 20]$. In their experiments, they specifically selected a value of m_{slic} equal to 10, which they found to offer a good balance between colour similarity and spatial proximity.
Euclidean distances in CIELAB colour space are visually meaningful for small distances. Outweighing pixel colour similarities (m_{slic}) prevents the spatial pixel distances from exceeding this perceptual colour distance limit. Hence, instead of using a simple Euclidean distance in the 5D space, distance measure D_s from each pixel j to the cluster center i defined as follows:

$$\begin{aligned} \mathrm{d_{color}} &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ \mathrm{d_{Spatial}} &= \sqrt{(x_{p_j} - x_{p_i})^2 + (y_{p_j} - y_{p_i})^2} \\ \mathrm{D_s} &= d_{color} + \frac{m_{slic}}{S} d_{Spatial} \end{aligned} \tag{3.1}$$

where D_s is the sum of the lab colour space distance and the spatial distance normalized by the grid interval S.

Enforcement Connectivity procedure:

After superpixel segmentation, SLIC uses a method to enforce all superpixels to be connected and prevent too small areas or any islands or disconnected area, which is called *Enforcement Connectivity* procedure. This method leads to non constant numbers of created superpixels making them unsuitable as direct CNN input model. This

inconsistency in number of created superpixel would cause trouble for our proposed approach. To prevent this problem, we used a modified version of SLIC. This version changes the connectivity enforcement algorithm provided by SLIC to keep the number of superpixels constant. This help us to provide the same size of input data for our network while keep the strength of SLIC method for having almost homogeneous superpixels.

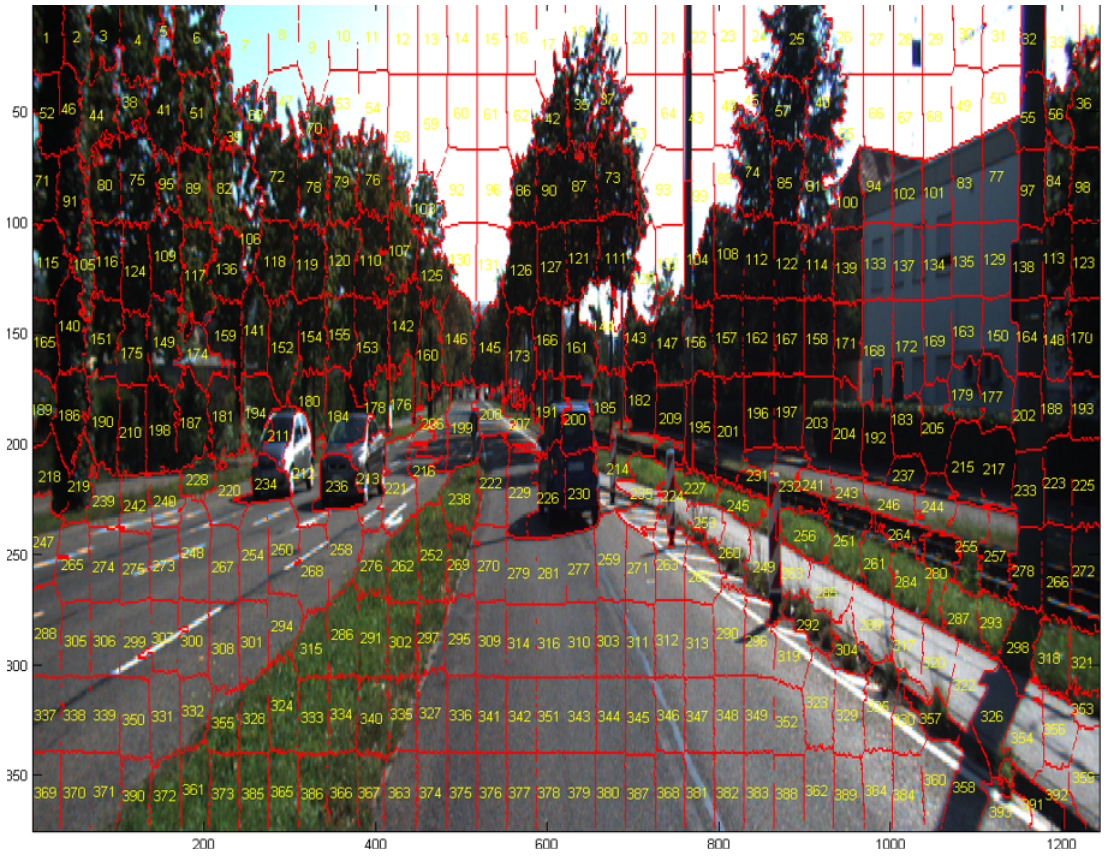
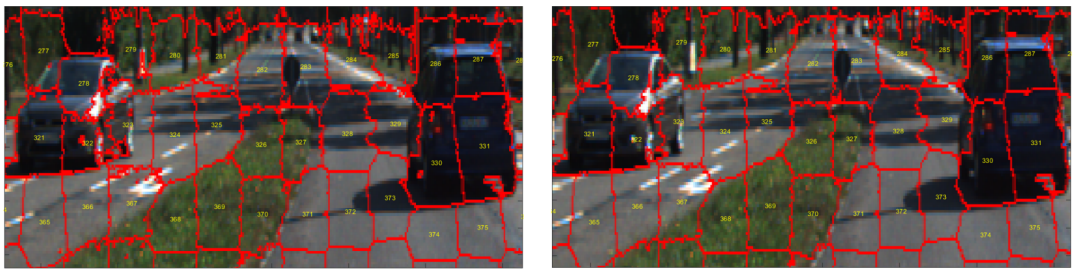


Figure 3.2: Superpixel segmentation of a sample image based on SLIC method after applying "Enforcement Connectivity" procedure.

Adapted SLIC Method

In the modified version, we perform following steps for each superpixel. First we find the whole set of adjacent superpixels, and the label-connected components in 2-D for each superpixel. Then if a certain superpixel has more than one segment with the same label, we keep the larger one and merge the rest into the nearest superpixel, which is picked up from neighbourhood(for example superpixels number 321 or 322 in Figure 3.3). The nearest superpixel is computed based on euclidean distance between the center of sub-segment to the center of each adjacent segment. Contrary to the

original version we do not remove any too small region with only one label-connected area. Figure 3.3 compares superpixel regions before and after applying of our enforce connectivity algorithm.



a) Before b) After

Figure 3.3: Enhancing superpixel segmentation through connectivity enforcement: Before and After. Disconnected superpixels (e.g., superpixels 321, 322 or 330) are intelligently integrated into their nearest neighbourhood, resulting in smoother segmentation boundaries.

The necessity to having a regular topology to be able to convolve the input data with kernels, motivated us to propose a superpixel lattice projection. The lattice is centered in the rectangular structure extracted from the first iteration of SLIC method (defined by the centers of the superpixels). This grid is directly used to establish a regular topology for the final superpixels, i. e. the superpixels generated by the last iteration step.

3.3.2 Hand-Crafted Feature Extraction

Feature selection acts as a preprocessing step that enables us to model relevant object characteristics in the image. We tested different combinations of features and decided on a particular combination, which gives the best performance. We considered three different feature groups, which are discussed in the following.

Colour Feature

Colour is the perceptual response of detectors such as the human brain or a camera to the light emitted or reflected by objects in an environment. It serves as a means by which important visual information is extracted and interpreted. It is widely recognized as one of the most informative features utilized by the human visual system for comprehending objects and scenes. This perception is influenced by several factors: a) the ambient light present in the environment, b) how objects reflect light within the scene, c) the human eyes or sensors that receive the light, and d) the subsequent processing of the received light [Weller and Westneat, 2019]. A mathematical model, that maps perceived colours onto a system of coordinates is called

colour space. Some of the colour space are: RGB (Red, Green, Blue), HSV (hue, saturation, and value), CMYK (cyan, magenta, yellow, and key), and CIELAB (also known as Lab). In CIELAB, "L" represents perceptual lightness, while "a" and "b" represent the four unique colors of human vision: red, green, blue, and yellow. Each color space has its own strengths and is designed to address specific colour-related concerns and objectives in research. A comprehensive study on the colour theory and colour space is beyond the scope of this thesis. We investigated the respective colour spaces and their usefulness [Garcia-Lamont et al., 2018], which prompted us to use them as handcrafted features in this work . We used different colour spaces RGB, Lab, HSV and computed the average values of all pixels within each superpixel for each colour channel separately.

RGB provides a computationally manageable format for storing images and are largely optimized for digital displays. However, it is not perceptually uniform and can display a smaller range of colours than the human eye can perceive. **HSV**, in contrast, attempts to more closely mimic human colour perception. However, HSV is non-linear and highly device dependent, which requires additional information about lighting in a scene, or the type of the sensor that capture the scene. In addition the separation of the luminance component makes it useful for image segmentation analyses and other applications. The **CIELAB** colour space, or simply *Lab*, is a colour model developed by the International Commission on Illumination (CIE). It is designed to represent all perceivable colours and provide a uniform colour space that is independent of devices or colourant systems. The CIE Lab space consists of three components: L, a, and b. The L component represents the lightness of the colour, ranging from black ($L = 0$) to white ($L = 100$). The a component represents the position along the green-red axis, with negative values indicating green and positive values indicating red. The b component represents the position along the blue-yellow axis, with negative values indicating blue and positive values indicating yellow. It was designed to make visual of an image closer to perception of the human eye along by decoupling the brightness information (L channel) from the chroma information (channels a and b) and takes into account some image enhancement in the colour space.

Defining descriptors in different colour spaces usually improves the description of object and texture image categories [Verma et al., 2010]. They are more robust against image variations such as lighting changes, rotation, and occlusions [Burghouts and Geusebroek, 2009].

Position Feature

Generally, road area can be detected from its surroundings based on the colour feature, however the appearance of shadows or similar pattern to the roads like side-walks, leads to the relatively difficult adequate prediction. As the colour to class distribution may vary for different positions and road is typically located in the bottom, we considered "position" as a second type of representative feature addition to the colour

feature. The average of vertical coordinates of pixels in each superpixel is selected as the location feature.

Texture Feature

Some local information like texture and shape can contribute to object and scene image classification. Investigation of our images represent significant changes between the texture of a road and its surroundings, especially compared to houses and trees. Roads tend to be flat and smooth, whereas trees and houses have more complex and compound textures. We use Local Binary Patterns (LBP)[Ojala et al., 1994] to computes correlation and disparity among pixels inside each superpixel. LBP showed to be promising for recognition and classification of texture images. It has important advantages such as rotation and gray-scale invariance. The original LPB operator is defined as a 3×3 window, where the central pixel is considered as a threshold, subtracted from the grey value of the adjacent 8 pixels. If the resulting value is negative, the pixel is set to '0', otherwise it is set to '1' which concatenate together to give an 8- bits code corresponding an integer ranging from 0 to 255 (a total of 256 types).
The basic LBP operator [Ojala et al., 1996] has a limitation in capturing dominant features with a large-scale structure due to its small neighbourhood size. This limitation arises because the basic LBP operator considers only a small group of neighbouring pixels, which may not effectively represent larger patterns or structures in an image. To overcome this limitation, an extension to the LBP operator was introduced [Ojala et al., 2002], which allows for a more flexible definition of the neighbourhood, denoted as (P, R). In the (P, R) neighbourhood notation, P represents the number of sampling points around the central pixel, and R denotes the radius or distance from the central pixel to the neighbouring points. Unlike the fixed circular neighbourhood of the basic LBP, the (P, R) neighbourhood enables the consideration of different radii and numbers of sampling points. This flexibility allows for capturing textures with varying scales and structures. By adjusting the values of P and R, the extended LBP can adapt to different texture patterns present in an image. A more formal description of the LBP operator can be given as:

$$LBP(x_{p_c}, y_{p_c}) = \sum_{n=0}^{P-1} \tau(I_n - I_c) \cdot 2^n \tag{3.2}$$

with P is the total number of neighbouring pixels considered around the central pixel, (x_{p_c}, y_{p_c}) is central pixel position with intensity I_c, and I_n is the intensity of the nth neighbouring pixel. τ is the threshold function defined as:

$$\tau(x) = \begin{cases} 1, & \text{if } x >= 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Another extension of LBP introduced the concept of uniform patterns [Ojala et al., 2002]. A Local Binary Pattern is considered uniform if it exhibits at most two bitwise transitions from 0 to 1 or vice versa when the binary string is considered circular. For example, patterns like 00000000, 11111111, 01111110, and 11001111 are classified as uniform, while patterns like 10101010 and 00110011 are non-uniform. To reduce the number of distinct patterns, a mapping table is utilized, assigning a unique label to each uniform pattern while grouping non-uniform patterns under a common label. Once the patterns are converted into uniform patterns, the feature vector for a single cell is constructed by counting the occurrences of each uniform pattern within the neighbourhood. This process significantly reduces the number of distinct patterns from 256 to 59. This reduction in length from 256 to 59 enhances the computational efficiency, memory usage, and reduces the risk of over-fitting in subsequent analysis or machine learning algorithms.

In conclusion, considering all the aforementioned features, a high-dimensional feature descriptor is defined for each superpixel. The image descriptor encompasses a total of 69 image features (see Figure 3.4), which include 9 color features (combining RGB, HSV, and Lab), 1 position feature, and 59 LBP features. This comprehensive feature set ensures high levels of accuracy and reliability in the analysis. The resulting data model is then inputted into a straightforward convolutional network, as illustrated below.

3.3.3 Network Architecture and Hyper-parameters

Contrary to most state of the art CNN-based semantic segmentation approaches, our proposed method does not require a complex network architecture to handle large image context, due to the pre-segmentation which improves computational time. Figure 3.5 shows our proposed CNN architecture. This network consists of two convolutional layers, two channel-wise feature layers (1×1 convolution filter) and one drop-out layer with non-linear activation function after each convolutional and channel-wise feature layer. The input of our method is defined by the superpixel lattice (see section 3.3.1) on each image with size of H_{img}/S and W_{img}/S, where S is initial superpixel side length and W_{img}, H_{img} are image width and height. The output is a set of three numbers, arranged in the superpixel lattice, to indicate respectively which of the three classes of the *road*, *non-road* or *un-labelled* they belong to. We explain them more in detail in section 3.5.

The weights in the first convolutional layer are initialized randomly from a Gaussian distribution with setting bias to zero. In our network, we employ Softmax loss (refer to section 2.5.5 equations 2.22 and 2.23) as the chosen loss function. Softmax loss, also known as Log Loss, quantifies the disparity between the predicted class probabilities (derived from the softmax function) and the actual class labels. For the final classification decision, our method selects the class with the highest predicted prob-

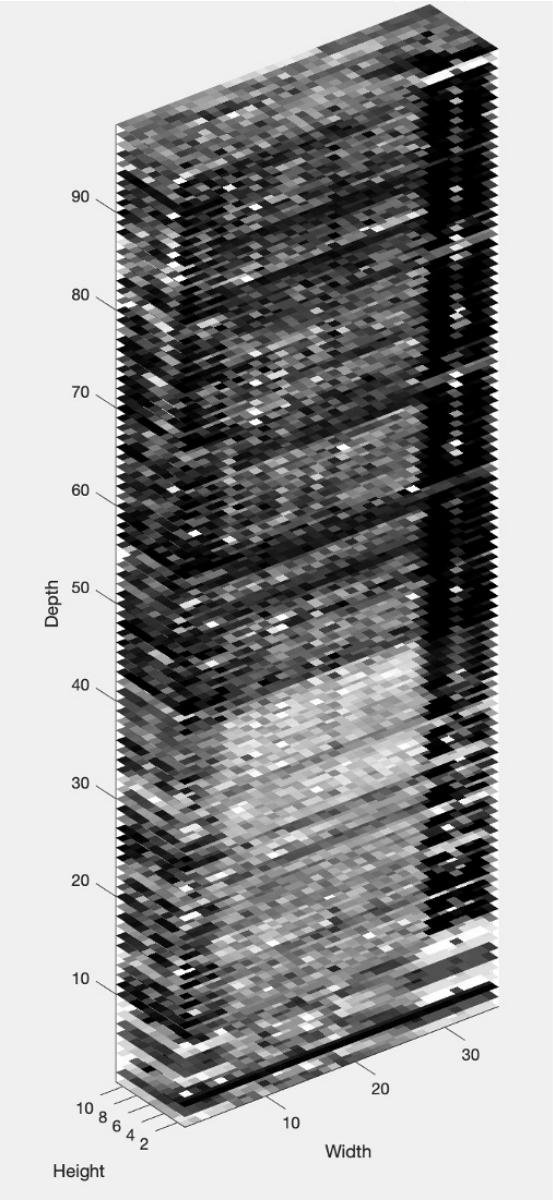


Figure 3.4: 3D Visualization of the Extracted Feature Representation for Superpixel-Based Segmentation. This figure illustrates the 69-dimensional feature representation derived from each superpixel, organized into a regular 11×36 grid. Each cell in the grid corresponds to a superpixel from the segmented input image, and the depth of the visualization represents the 69 different features extracted per superpixel. The features include colour, texture, and positional attributes, all of which are crucial for effective road semantic segmentation. This structured representation is used to facilitate feature learning in subsequent convolutional network processing.

ability for each pixel. This approach aligns with standard practices, where the class with the maximum softmax score is considered the most likely class for each instance. The training network undergoes optimization using the stochastic gradient descent (SGD) algorithm for a maximum of 300 iterations. In this optimization process, the current state of the model's error is estimated repeatedly based on the loss (objective) function. This estimation allows the weights to be updated and reduces the model's loss in the subsequent evaluation.
To ensure effective optimization, the learning rate and other hyper-parameters are fine-tuned on the validation set. In our case, the learning rate is set to 0.1×10^{-4}, and the weight decay is set to 0.0001. Additionally, we employ a momentum value of 0.9 and utilize mini-batches of size 50. All of the above parameters are carefully selected through empirical experimentation to achieve a reasonably good loss value and error rate. These parameter settings remain consistent across all datasets. Figure 3.6 presents the visual representation of our loss (objective) and error plots during the training phase. These plots provide valuable insights into the performance and progress of our model throughout the training process.

1. **Error log plot:** The error log plot provides a visual representation of the errors or misclassification made by our CNN model throughout the training process. It displays the training error and validation error on the y-axis, while the x-axis represents the number of epochs. By analysing the error log plot, we can gain insights into the convergence behaviour of our model. Ideally, we want both the training and validation errors to consistently decrease and converge to a low value. However, if the training error decreases significantly, while the validation error increases, it indicates over-fitting, which means our model becomes too specialized to the training data and struggles to generalize effectively to unseen examples.

2. **Objective Log Plot:** The objective log plot illustrates the trend of the loss function values (specifically, the softmax loss in our model) throughout the training epochs. This plot offers valuable insights into the optimization process. Similar to the error log plot, the y-axis represents the loss values, while the x-axis denotes the number of epochs. By analysing the objective log plot, we can evaluate the model's proficiency in optimizing the softmax loss function. Ideally, both the training and validation loss curves should exhibit a consistent decline and converge to a low value. However, if the training loss decreases significantly, while the validation loss increases, it indicates over-fitting.

The error log plot primarily focuses on the misclassification rate and generalization performance of your model, while the loss log plot emphasizes the optimization progress and model convergence based on the loss function. Both plots offer valuable insights, but they provide different perspectives on the model's performance and behaviour during training. Therefore, it is recommended to analyse and present both plots to gain a comprehensive understanding of the CNN model's training process in your thesis.

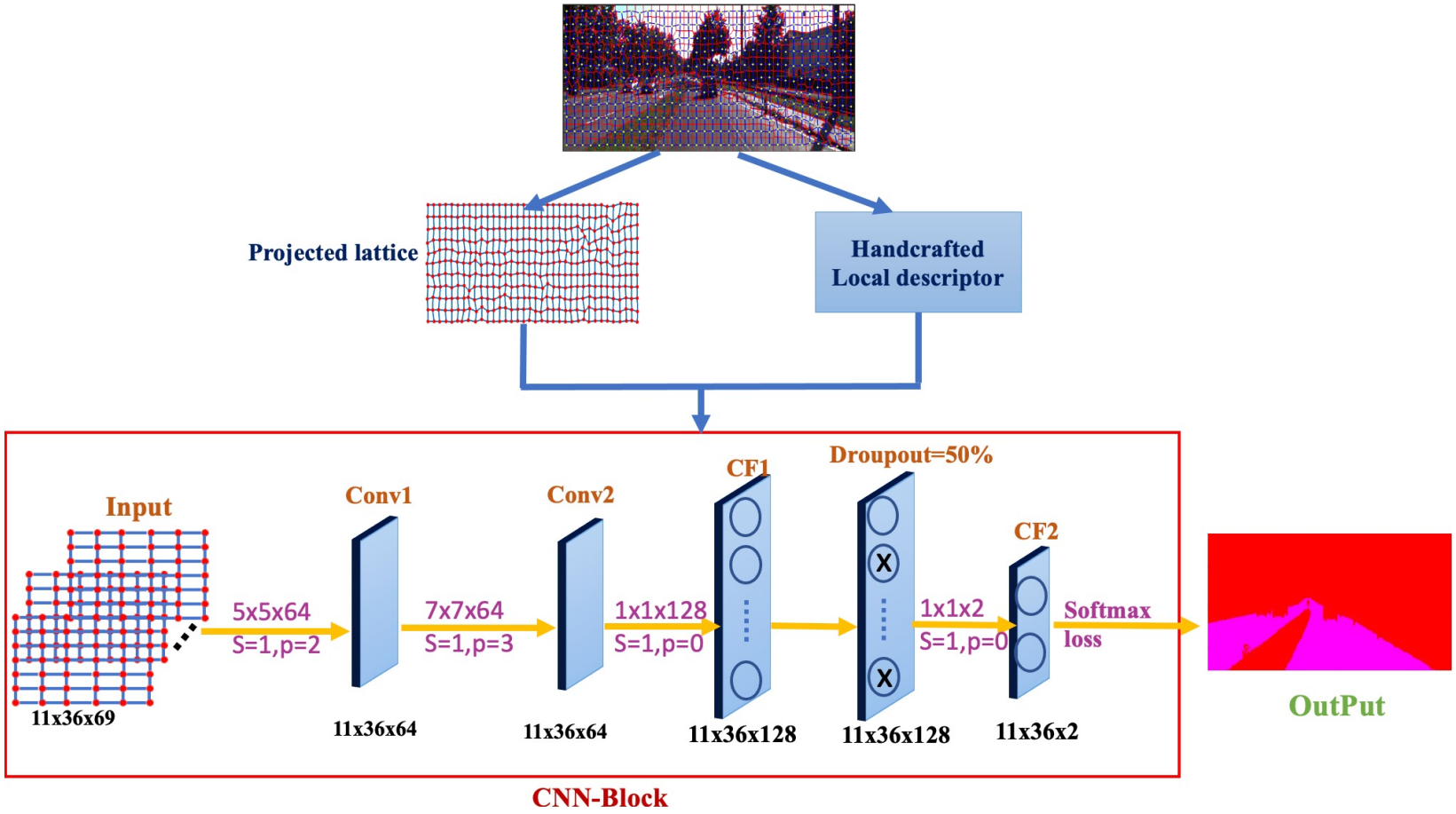


Figure 3.5: Our proposed CNN architecture based on superpixels and a high dimensional feature descriptor input data model. This figure shows the input data size, convolutional kernel size and the numbers of the output feature produced by each filter layer.

In the context of our training process, it is noteworthy that training was conducted for a total of 300 epochs. However, during the period from 200 to 300 epochs, an interesting observation emerges: while the training loss exhibits a stable behaviour, the validation loss demonstrates a slight increase. Notably, the error rates for both the training and validation sets exhibit a consistent pattern after the 200th epoch, indicating a lack of further improvement.

This behaviour suggests that our model's performance has reached a plateau or a state of near-optimality after 200 epochs of training. Despite the observed slight increase in the validation loss, the error rate remains constant, which indicates that the model is still achieving satisfactory results in terms of classification accuracy. This finding implies that our model has effectively learned the underlying patterns in the data to a significant extent during the initial 200 epochs of training. Consequently, subsequent adjustments or fine-tuning do not yield substantial improvements in the loss metric. This phenomenon aligns with the notion that our model's capacity to enhance its performance has been reached.
In the objective plot, the decline in the loss value reflects an improvement in the predicted probabilities as training progresses, particularly until the 200th epoch. This observation supports the idea that the initial phase of training is crucial for achieving considerable enhancements in model performance.
Overall, these findings emphasize the significance of the first 200 epochs in our training process, as they allowed the model to effectively learn the relevant patterns in the data. The subsequent epochs, while not leading to substantial improvements in loss, maintain the model's classification accuracy at an acceptable level. These insights contribute to our understanding of the model's capacity and its behaviour throughout the training process.

3.4 Implementation Details

We tackle the problem of the efficient semantic road segmentation by segmenting the urban scene images into superpixels, deriving road relevant features, and constructing a rational feature model using machine learning mechanism to segment road object in the images. The images are broken down into small units by extracting superpixels from the original image using the Simple Linear Iterative Clustering (SLIC) algorithm of section 3.3.1. The main features we extracted from the superpixels are colour, texture and location according to section 3.3.2. Then, we compare two different machine learning models a) Supporting Vector Machines (SVM) [Cortes and Vapnik, 1995] and b) Convolutional neural network (CNN) models based on the generated features. SVM is used to analyse the applicability of the proposed input data model to obtain an efficient semantic segmentation result and investigate how much deep learning could empower our proposed method. Before we describe the proposed method in detail, we discuss the two main public datasets for urban scene images which are used as the input datasets in this work.

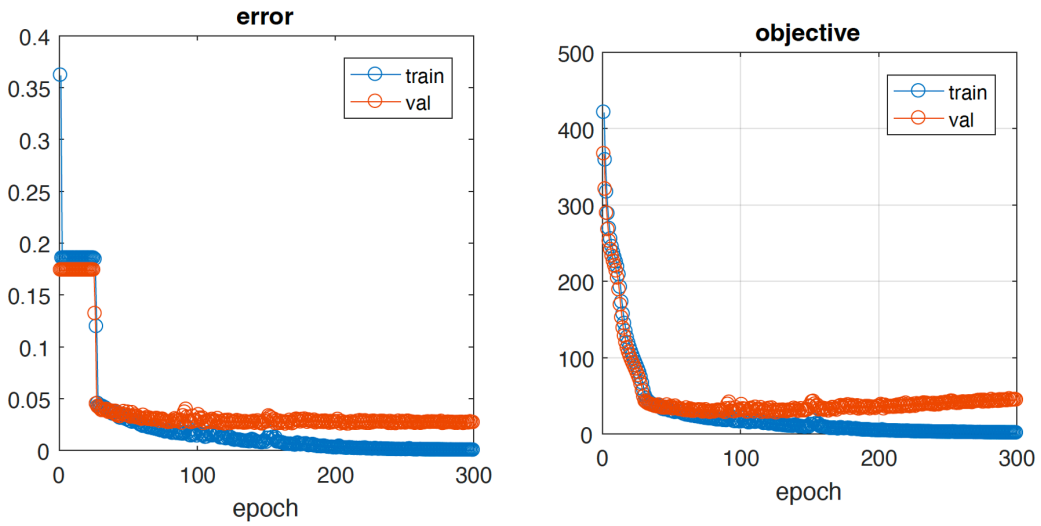


Figure 3.6: The objective and error plots during the training. The training error is depicted by the blue line and the validation error is represented by an red dotted line. Softmax loss is used as objective function and stochastic gradient descent as the optimization algorithm. The plot shows that the training process converged well. The plot for loss is smooth, given the continuous nature of the error between the probability distributions. Log Loss(softmax loss) gradually declines as the predicted probability improves after almost 200 iterations. After 200 epochs, the learning ability of the network is almost steady or slightly getting overfit.

3.4.1 Datasets

We evaluate our method using two widely-used challenging datasets comprising urban scenarios, i.e. the KITTI [Fritsch et al., 2013a] and the Cityscapes [Cordts et al., 2016] datasets. In the following we give a brief description of the datasets followed by the evaluation results.

KITTI Dataset

KITTI *road* estimation data set comprises 502 8-bits RGB images splits in train, validation and test sets with ground truth label for three semantic classes. KITTI is one of the most popular datasets for road segmentation in urban scene applications. The training set comprises a total of 289 (URBAN) images, which are further categorized into three sets, each representing a specific road scene category commonly found in inner cities. Specifically, there are 95 images with urban markings (UM), 96 images with multiple marked lanes in urban areas (UMM), and 98 images depicting unmarked urban streets (UU). In the UM dataset, annotations for the ego-lane are also provided.
The test set comprises a total of 290 (URBAN) images, with 96 images showcasing

urban markings (UM), 94 images depicting multiple marked lanes in urban areas (UMM), and 100 images of unmarked urban streets (UU). However, it is important to note that the ground truth for the test set is not publicly available.
The image dimensions vary with the width lying in $[1226, 1238, 1241, 1242]$ and their height in $[370, 374, 375, 376]$. We selected 20% of train set images from 3 different categories UM,UMM,UU for the validation set. These images are completely from different video sequences which are not part of the train set.
In evaluating road and lane detection performance on the KITTI dataset, models are analysed within the Birds-Eye-View (BEV) space, with their outputs rendered as confidence map or binary map that delineate road areas. This evaluation framework employs core parameters such as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These parameters respectively quantify the number of road pixels accurately predicted, non-road pixels correctly identified, non-road pixels erroneously classified as road, and road pixels incorrectly marked as non-road.
The assessment leverages several metrics to evaluate model performance comprehensively. Accuracy (ACC) measures the proportion of correctly identified pixels across the dataset, reflecting the overall correctness of the model's predictions. Precision (PRE) assesses the ratio of correctly predicted road pixels against all pixels labelled as road by the model, indicating the precision of the model in identifying road areas. Recall (REC) captures the fraction of actual road pixels that the model correctly identifies, reflecting the model's sensitivity to road features. The False Positive Rate (FPR) and False Negative Rate (FNR) provide further insights into the types of errors made by the model, indicating the proportion of non-road pixels misclassified as road and the proportion of road pixels missed by the model, respectively.
A key focus of our evaluation is the Maximum F-measure (MaxF) [Everingham et al., 2010], which here represents the highest F1 score achievable by the model across various decision thresholds. This metric underscores the optimal balance between precision and recall, essential for accurately classifying road pixels. Notably, in our method, MaxF is directly derived from the binary map outputs without adjusting decision thresholds, highlighting the model's inherent capability to delineate road areas effectively.
Additionally, Average Precision (AP) [Everingham et al., 2010] is an essential metric in object detection that encapsulates a model's precision across different recall levels (r). AP computes the average of the highest precision values p for any recall $\tilde{r} \geq r$, across eleven evenly spaced recall thresholds from 0 to 1, at 0.1 intervals. This approach captures the model's efficacy on the precision-recall curve, providing a unified perspective on its performance. The equations for these metrics, adapted from [Fritsch et al., 2013a], are delineated in Equations 3.4 to 3.10, providing a mathematical foundation for our evaluation methodology.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

$$PRE = \frac{TP}{TP + FP} \tag{3.5}$$

$$REC = \frac{TP}{TP + FN} \tag{3.6}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.7}$$

$$FNR = \frac{FN}{TP + FN} \tag{3.8}$$

$$F1 = 2 \times \frac{PRE \times REC}{PRE + REC} \tag{3.9}$$

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{3.10}$$

In our experimental setup using the KITTI dataset, we carefully selected the parameters for the SLIC algorithm. Specifically, we set the number of superpixels (N_{sp}) to 400 and the weight (m_{slic}) to 35. As a result, each image was divided into 396 superpixels, which were then projected onto an 11×36 lattice to serve as the input for the CNN.

Cityscape Dataset

Cityscapes is another new large dataset for semantic scene understanding in urban environments [Cordts et al., 2016], comprising the images, which were taken under good or medium daytime weather conditions and their corresponding pixel-wise ground truth label for 19 semantic categories consists of road (including road and side-walk), car, pedestrian, bicycle, etc. One void class is considered for pixels not belonging to the nineteen classes. The dataset contains 2975 training, 500 validation, and 1525 test images. All of the images in this dataset are in the same size of 1024×2048 pixels.

In the Cityscape dataset [Cordts et al., 2016], the standard Jaccard Index [Everingham et al., 2015], commonly referred to as the PASCAL VOC intersection-over-union

(IoU) metric, is employed to evaluate performance on test set. The IoU is calculated as the ratio of true positive (TP) pixels to the sum of TP, false positive (FP), and false negative (FN) pixels, across the entire test set. A comprehensive explanation will be provided in Section 3.5. There are two separate mean performance scores in Cityscape benchmark: *IoUcategory* and *IoUclass*, which address the two semantic granularities classes and categories respectively. In each of them, pixels labelled as void do not contribute to the score.

To conduct our evaluation, we present the results obtained from the validation and test sets. Our focus during evaluation was solely on road segmentation. Consequently, we modified all labels, except for the road class, to be classified as the background label. This enabled us to exclusively assess the performance of our approach in terms of road segmentation. In our Cityscape experiment, we conducted the analysis by selecting specific SLIC parameters: $N_{sp} = 2000$ for the number of superpixels and $m_{slic} = 10$ for the weight. These parameter choices were applied to half-resolution images (512×1024), resulting in 2048 superpixels for each image. Subsequently, the superpixels were projected onto a 32×64 lattice, which served as the input data for the CNN in our experiment.

3.4.2 Model Analysis

To address the challenge of achieving both fast and accurate road segmentation, we propose an innovative approach that leverages super-pixel segmentation as the input data model, combined with Convolutional Neural Network (CNN) as the final decision for choosing the superior machine learning approach for our semantic segmentation task.

As mentioned in section 2.7, conventional segmentation approaches commonly employ two strategies decomposing each image frame into fixed-size patches or analysing individual pixels throughout the entire image. However, pixel-level classification is prone to noise, which poses difficulties in accurately identifying homogeneous regions and requires substantial computational resources and powerful GPUs. While fixed-sized patches offer noise reduction and improved information aggregation, they suffer from a lack of adequate spatial context. This limitation gives rise to suboptimal pairings in nearest neighbour searches and compromises the accuracy of the segmentation results. Moreover, these patches can encompass multiple distinct image regions, leading to a degradation in the overall classification performance.
In this study, two distinct machine learning approaches are employed for semantic road segmentation using the KITTI dataset. The first approach involves training a support vector machine (SVM) directly on the inhomogeneous superpixels extracted through the SLIC method. Additionally, handcrafted features, such as colour, position, and texture, are extracted from the labelled superpixels. These features are then utilized as input for the SVM to classify the road objects in the images.
In contrast, the second approach introduces a novel method based on Convolutional Neural Networks (CNN). Instead of directly using the inhomogeneous superpixels

as input, this approach utilizes a mask that covers the irregular superpixel. This mask is specifically designed to provide the necessary information for the convolution operation within the CNN. By incorporating this mask-based approach, the CNN effectively addresses the challenges posed by irregular superpixels and enhances the segmentation performance for road object.

SP-SVM

Support Vector Machines (SVM) [Cortes and Vapnik, 1995] is a discriminative classifier, which maximize the margin between data near the classification boundaries. A support vector machine (SVM) classifier was trained on samples of road images based on superpixel extraction. First, each image from KITTI dataset with size $H_{img} \times W_{img}$ is segmented into superpixels by SLIC method [Achanta et al.,], where the number of desired superpixel and compactness are chosen as in the CNN-based method (respectively $N_{sp} = 400, m_{slic} = 35$). Contrary to CNN's, that require 2D input data, SVM's generally need a vector (1D input data). Hence, in this approach, we do not need the projection lattice (like our proposed CNN method). Due to the different size of the images in the KITTI dataset, discussed in section 3.4.1, the total number of generated superpixels varies between $[396 - 400]$ superpixels for each image.
We assign the majority label of all pixels in each superpixel as corresponding class label to that superpixel. Each superpixel is described by 69 features includes, a set of the colour channels in different colour space, position and the Local Binary Patters (LBP) 3.3.2. We provided a vector array of all superpixels from all images in the training set and the corresponding vector array of their labels as data input model. The classifier was implemented in Matlab 2015b using LibSVM library. We used Linear kernel provided in the this library. We did our experiment first with binary SVM for two classes (road and un-road when the side-walk is excluded from the images). By applying this binary classification on the validation set with 10 images, we achieved road segmentation accuracy of about $87,81\%$. Next time we applied Multi-SVM with 3 different classes: road, un-road and side-walk and achieved 94.34% of the road segmentation accuracy. The results are shown in figure 3.7. Additional examples are illustrated in Appendix I.

Because SVM only cares about samples near the margin, it usually avoids the problem of over-fitting, however the major downside of SVM is that it can be painfully inefficient to train compared to other powerful models like CNN. Although it is good, it is certainly not the best.
Using linear SVM make us sure that if this linear superpixel based binary(multi) classifier can segment the road in the images almost good, a powerful machine learning approach like CNN should provide better segmentation results. It is important to note that, While an SVM-based approach was initially explored as part of this study to analyse the potential effectiveness of our proposed method, a direct comparison between the SVM and CNN results is not presented in this section. This decision is based on several factors. Firstly, the SVM approach primarily served as a preliminary step to assess the feasibility of the subsequent CNN-based approach. Secondly,

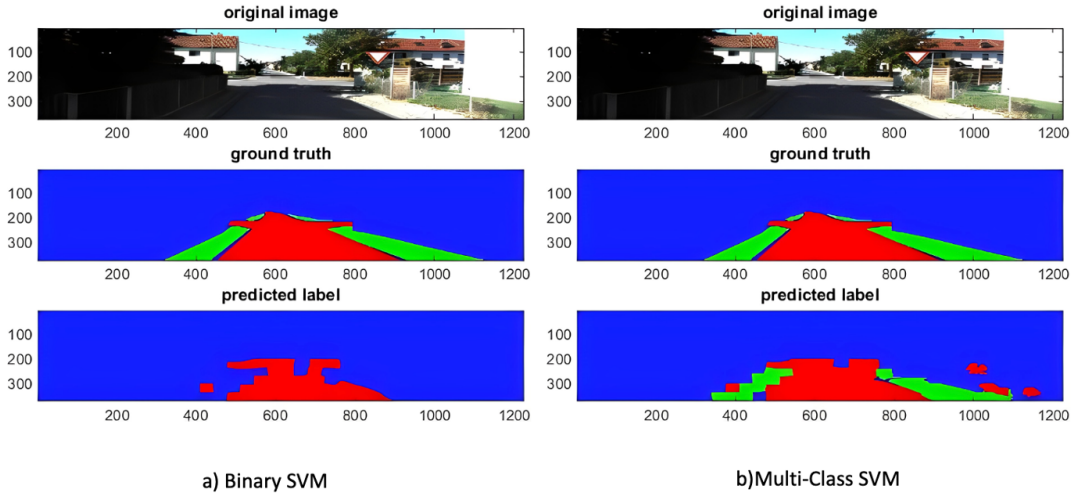


Figure 3.7: Road segmentation with SVM. a)is the result obtained from the binary SVM, b) shows the Multi-class SVM result.

the SVM-based evaluation was conducted on a subset of the training and validation images from the KITTI dataset, which may not provide a fair basis for direct comparison with the CNN results obtained on the entire dataset. Lastly, due to hardware constraints and availability, a comprehensive comparison of computational efficiency between the two methods, in terms of running time, was not conducted. Therefore, the section 3.5 focused on the analysis and discussion of the CNN-based approach performance, which constitutes the main contribution of this study.

SP-CNN

We address the use of superpixels as the visual primitives for semantic segmentation. superpixels are obtained from an over-segmentation of the image and they aggregate visually homogeneous pixels, while respecting natural terrain boundaries. As discussed in part(3.3.1), we used SLIC method for superpixel segmentation. This method creates different size of irregular-shaped image segmentation, which are not suitable for convolution operation. Contrary to our SVM method, we need a regular topology to be able to convolve the input data with kernels. On one hand, employing regular superpixel segmentation methods helps preserve the topology of the image. However, it also leads to a loss of maximum homogeneity in terms of the texture within each superpixel.
To address this issue, we propose the use of a mask placed over the original superpixel segmentation. This mask is centered on a regular grid derived from the initial iteration of our Adapted SLIC method (see section 3.3.1). By incorporating this approach, we establish a grid structure, that facilitates the application of a convolutional layer, while simultaneously maintaining the correlation between pixels within each cluster. For a visual representation, please refer to Figure 3.8.
In this method we don't need exactly the same size of the images, but we need the

same total number of superpixels on each image. To keep the equality between the amount of the number of created inhomogeneous superpixels with the amount of the superpixels in the mesh grid, we used our Adopted SLIC method from 3.3.1. It is important, that the class labelling was determined based on the majority label of all pixel labels in the inhomogeneous superpixels, not in regular superpixels.
This lattice together with the high dimensional image descriptors are fed to our designated convolutional neural network described in section 3.3.3, to classify the superpixels of the image according to semantic categories. Figure 3.9 displays a simple visualization of a sample convolutional kernel and corresponding output feature map from a convolutional layer in our network. A two-dimensional filter learned by the model is visualized to discover the types of features that the model will detect in specific intermediate step, and the activation map output by convolutional layer can be inspected to understand exactly what features were detected for a given input image.

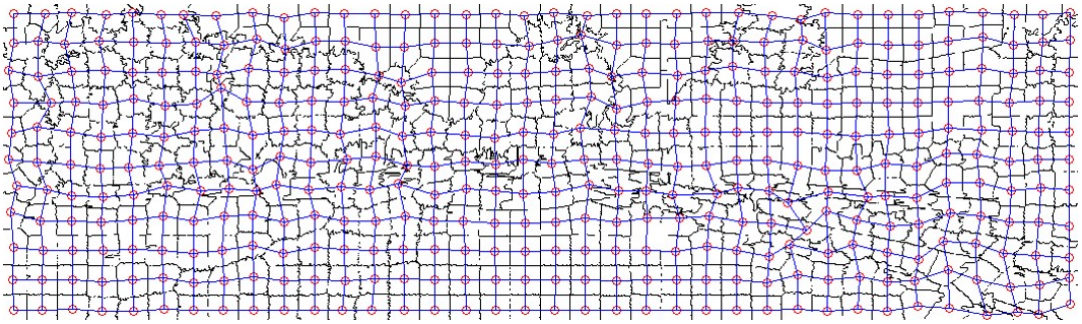


Figure 3.8: Grid scheme on top of superpixel segmentation (After) "Enforcement Connectivity".

3.5 Evaluation and Discussion

The training and testing phases were conducted using a hardware setup equipped with an *Intel(R) Core(TM) i7-4790K CPU* running at a clock speed of *4GHz*.

To ensure optimal performance, the feature vectors extracted from superpixels underwent a normalization process. This involved calculating the mean and standard deviation for each channel within the training set. Subsequently, these computed values were employed to normalize the feature vectors in both the training and evaluation stages.
Normalizing the feature vectors using the mean and standard deviation serves several purposes. Firstly, it helps center the data distribution around zero, which aids in stabilizing the learning process and improving convergence during training. Secondly, it enables fair comparisons and consistent results by eliminating any potential bias introduced by varying scales or offsets in the original feature values.
It is essential to perform the normalization solely on the training set and not on

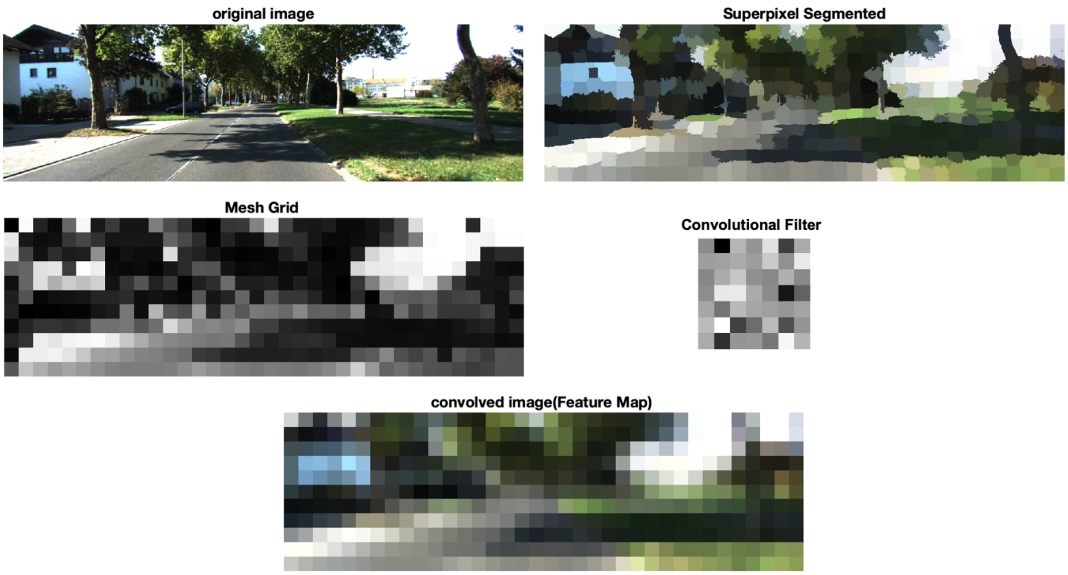


Figure 3.9: Visualizing a Filter and corresponding output Feature Map from our Convolutional Neural Network. First superpixel segmentation is applied on original image to extract the required high dimensional feature descriptor. Then it is projected on regular mesh grid to be able to do the convolutional operation. A sample kernel filter from our network is convolved with this lattice and results the corresponding feature map.

the combined train, validation, and test sets. This approach ensures that the normalization process remains unbiased and reflective of the training data's statistical characteristics. By applying the mean and standard deviation computed from the training set to the evaluation phase, we maintain consistency and allow the model to generalize effectively to unseen data.

To ascertain consistency in the comparison with other state-of-the-art methods using the KITTI dataset [Fritsch et al., 2013a], we employed the evaluation metrics specified by the KITTI benchmark, as discussed in Section 3.4.1. By utilizing these evaluation metrics, we maintain the standard framework for assessing the performance of our method in relation to others.
Our primary objective was to perform road segmentation from the background. To assess the effectiveness of our proposed approach, we conducted evaluations in two domains, a) the native pixel grid and b) the superpixel grid. These evaluations encompassed two perspectives:

- The Image perspective (egocentric view), which reflects the scene as observed from the viewpoint of the camera or the person capturing the images.
- The Birds Eye View (BEV) projection provided by the KITTI dataset, as detailed in section 3.4.1.

In the superpixel domain, the evaluation process involves defining the ground truth for each superpixel based on the majority of pixel labels contained within it. If the majority does not exceed 80%, we assign a new label called *Un-labelled* to these instances, resulting in a total of three classes *road*, *non-road*, and *unlabelled*. However, since the KITTI road benchmark evaluates only two classes (road and non-road), we modified the labelling of superpixels in the test set by reassigning them based on the majority value.

3.5.1 KITTI Evaluation on Image Perspective

We adhered to the evaluation methodology outlined in the KITTI dataset (see [Fritsch et al., 2013a]). Due to the unavailability of ground truth for the test data, we utilized the validation set to assess our approach from the image perspective. In Figure 3.10, we present two illustrative results obtained from the KITTI dataset, demonstrating the successful segmentation of the road. Additional examples are available in Appendix I.

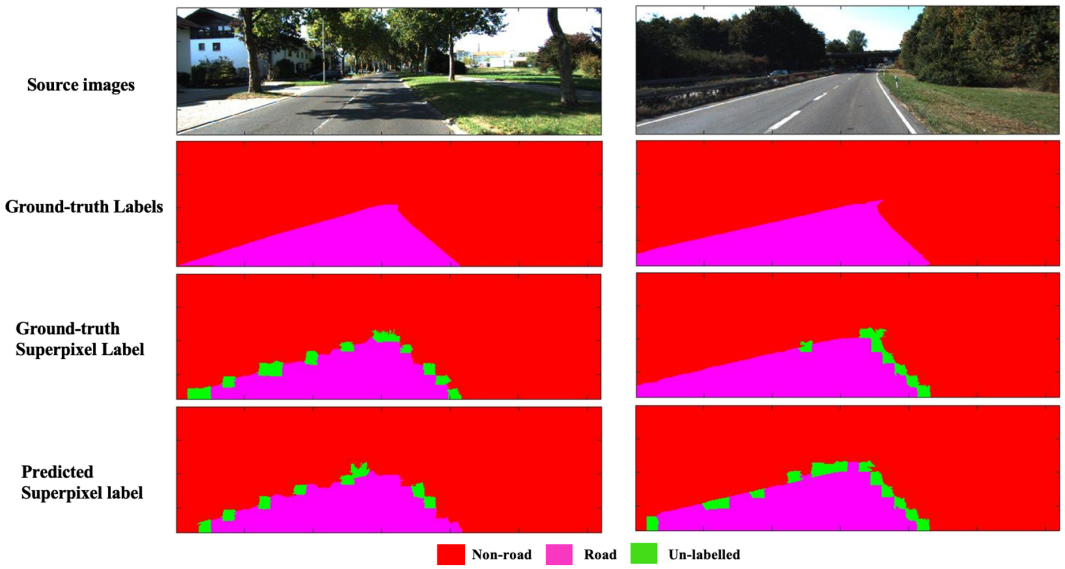


Figure 3.10: Road segmentation based on proposed SP-CNN method for two samples from KITTI. The first horizontal row shows input images. Ground-truth based on pixels and superpixels are shown in the second and third rows. The last row shows the predicted results based on superpixel labelling

Table 3.1 presents the average evaluation results obtained from the validation set for various urban categories in the KITTI dataset, considering both the superpixel and pixel domains. Our proposed model and parameters, as described in sections 3.4.2 and 3.3.3, were utilized to evaluate the results for each dataset based on both superpixel and pixel ground-truths. In this evaluation, the positive class was defined as

"road", while the negative class consisted of "non-road" and " unlabelled" instances. Furthermore, we also included the evaluation results obtained from the Cityscape validation dataset (further discussed in section 3.5.3). To ensure a consistent comparison between the KITTI and Cityscape datasets, we utilized the evaluation metric from KITTI to evaluate the Cityscape validation set as well.

Dataset	**B**enchmark	**A**CC	**M**axF	**P**RE	**R**EC	**F**PR	**F**NR
KITTI	Superpixel	97.37%	92.57%	94.21%	91.62%	5.79%	1.86%
	pixel	96.50%	90.08%	92.80%	88.07%	7.20%	2.65%
CityScape	Superpixel	95.75%	92.44%	89.08%	96.76%	10.92%	1.14%
	pixel	94.10%	90.01%	84.41%	97.14%	15.59%	1.89%

Table 3.1: Evaluation results on KITTI and Cityscape validation sets. For the evaluation of the following experiments we used these metrics: accuracy (ACC), maximum F-measure (MaxF) or F1, precision (PRE), recall (REC), false positive rate (FPR), false negative rate (FNR)

The evaluation results on the KITTI dataset, presented in Table 3.1, offer valuable insights into the performance of our road segmentation approach in different categories. In the superpixel domain, our method achieved an impressive average accuracy (ACC) of 97.37%, indicating a high level of correct road predictions. This was further supported by a substantial F-measure (MaxF) of 92.57%, reflecting the balance between precision (PRE) and recall (REC) measures. The precision score of 94.21% highlights the model's ability to accurately classify road pixels, while the recall score of 91.62% demonstrates its effectiveness in capturing a significant portion of the road pixels present in the images.
Furthermore, the low false positive rate (FPR) of 5.79% signifies the model's capability to minimize the misclassification of non-road pixels as road, which is crucial for avoiding false detections. Additionally, the low false negative rate (FNR) of 1.86% indicates the model's proficiency in capturing the majority of road pixels, reducing the likelihood of missing relevant road segments.

In the pixel domain, our approach achieved a slightly lower but still impressive average accuracy of 96.50%. This indicates a strong performance in accurately classifying road pixels at the individual pixel level. The F-measure of 90.08% suggests a good balance between precision and recall, with a precision score of 92.80% and a recall score of 88.07%. These results highlight the model's ability to accurately identify road pixels while maintaining a relatively low rate of false positives and false negatives.

Comparing the results between the superpixel and pixel domains, we can observe that the superpixel-based approach consistently achieves higher accuracy, F-measure, and recall scores compared to the pixel-based approach. This issue arises because the final label assigned to each superpixel is determined by the majority pixel label within it. Consequently, when superpixels lack homogeneity, this assignment based on majority voting may affect the segmentation result at the pixel level, potentially introducing some inaccuracies. However, it does not significantly impact the overall superpixel-level results, as the majority label within each superpixel tends to dominate. Despite these challenges, the pixel-based approach demonstrates commendable performance, showcasing its ability to classify road pixels at a finer granularity, while still achieving competitive evaluation scores.
Overall, these evaluation results on the KITTI dataset demonstrate the effectiveness of our proposed road segmentation approach. The high accuracy, F-measure, precision, recall, and low false positive and false negative rates indicate the model's robustness in accurately identifying road pixels, thereby contributing to the reliable detection and understanding of road scenes.
Although the road segmentation in Figure 3.10 shows overall success, it encounters challenges near the road border, resulting in false detections. The more detailed view in figure 3.11 reveals, this issue arises from the inherent nature of superpixel segmentation, which struggles to accurately identify the precise boundary of the road. Superpixels are formed to group similar pixels, often encompassing both the road and its surroundings. As a consequence, the compactness of the superpixels leads to false detections at the road border. Additionally, the presence of shadows further complicates the segmentation process. To address these challenges, a refinement step will be introduced in the subsequent chapter to enhance the accuracy and localization of the road segmentation.

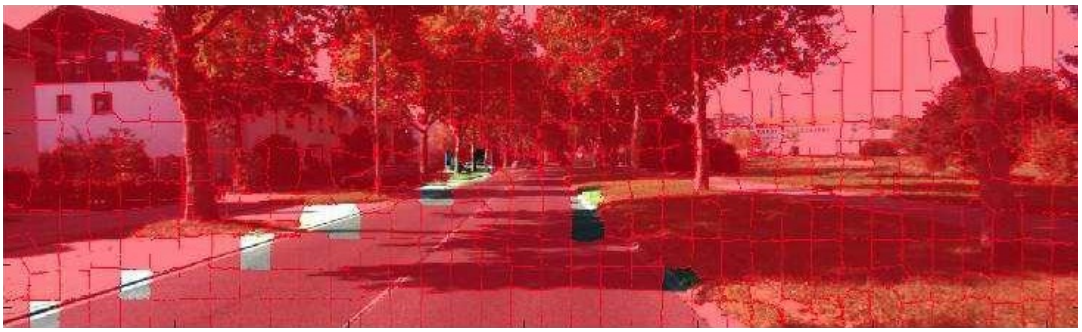


Figure 3.11: Wrong-predicted area is shown as non-red superpixels

Evaluation between irregular and regular superpixels

In our subsequent experiment, we demonstrate the superiority of well-homogeneous irregular-shaped superpixels, generated through our customized superpixel creation method, over regular-shaped fixed-size patches. The irregular-shaped superpixels exhibit higher classification accuracy compared to the less homogeneous fixed-size patches, enabling more effective navigation in complex terrain environments.
In this experiment, we employed a regular superpixel segmentation technique to fit the superpixels into fixed-size patches. Specifically, we utilized the first iteration of our Adapted SLIC method on all images from the KITTI dataset, resulting in a regular grid-like structure with 396 superpixels per image. This approach ensured a consistent number of superpixels and preserved the correlation between pixels within each superpixel group based on the criteria discussed in section 3.3.1.
However, it is crucial to acknowledge that although the regular superpixel segmentation enforced a uniform shape for each superpixel, it did not guarantee homogeneity across the superpixels. Consequently, the superpixels within the grid structure might exhibit variations in terms of their internal homogeneity and may not accurately represent distinct objects or regions within the image.
Subsequently, we extracted the previously proposed feature vector (see section 3.3.2) from each patch obtained through the regular superpixel segmentation. It is worth noting, that this feature extraction approach differs from our SP-CNN techniques, where the feature descriptors are extracted from irregular superpixels instead of the lattice grid. This disparity in feature extraction from irregular superpixels and regular ones can introduce potential drawbacks, as regular superpixels may not fully capture the intricate details and nuances of the underlying objects or regions of interest in the image.

The evaluation results from Table 3.2 showcase the performance of the fixed-size superpixel method and the irregular superpixel method in road segmentation.The evaluation was conducted in both the superpixel and pixel domains.
At the superpixel level, the irregular superpixel method outperforms the fixed-size superpixel method in terms of F-measure. The irregular superpixel method achieves an F-measure of 92.57%, indicating a higher overall accuracy in segmenting road superpixels compared to the fixed-size method, which yields an F-measure of 87.52%. This suggests that the irregular superpixel method captures more precise and accurate boundaries of road segments within each superpixel.
Similarly, when examining precision, the irregular superpixel method achieves a higher score of 94.21%, compared to the fixed-size superpixel method with a precision score of 80.46% in superpixel domain. This signifies that a larger proportion of road superpixels identified by the irregular method are correctly classified as road segments. However, when considering the results at the pixel level, both the irregular superpixel and the fixed-size superpixel method exhibit a decrease in performance compared to the superpixel level. The irregular superpixel method achieves an F-measure of 90.08%, while the fixed-size superpixel method yields a slightly lower F-measure of 85.19%. This indicates that both methods encounter challenges in accurately seg-

menting individual road pixels. The relatively small discrepancy of 5% in the overall F-measure between the two methods is primarily influenced by the accurate identification of non-road regions rather than road segments. It is because of the imbalanced class distribution between road and non-road regions within each image. This imbalance affects the performance evaluation, particularly in terms of road segmentation. However, it also highlights the high correctness achieved in segmenting non-road areas, as indicated by the overall F-measure scores.
Despite the decrease in accuracy at the pixel level, the irregular superpixel method still demonstrates superior performance in precision. It achieves a precision score of 92.80%, whereas the fixed-size superpixel method obtains a precision score of 78.85%. This large gap of almost 14% of average precision implies the low ability of the road segmentation in patch-wise method. This gets even worse by existing shadow or obstacle on the road surface (See Figure.3.12).
Overall, the results indicate that the irregular superpixel method outperforms the fixed-size superpixel method in both superpixel-level and pixel-level road segmentation. The irregular method achieves higher accuracy, precision, and F-measure scores, indicating its effectiveness in capturing the boundaries of road segments more accurately. However, it is important to consider, that both methods experience some challenges in accurately identifying road pixels at the pixel level, suggesting the presence of potential misclassification or missed road pixels in both approaches.

Method	**B**enchmark	**M**axF	**P**RE	**R**EC
Fixed-size SP	Superpixel	87.52%	80.46%	98.07%
	pixel	85.19%	**78.85%**	94.70%
irregular SP	Superpixel	92.57%	94.21%	91.62%
	pixel	90.08%	**92.80%**	88.07%

Table 3.2: Evaluation results on both regular and irregular shape of superpixels on KITTI validation set.

An alternative approach is to use resized images as input instead of feeding the model with superpixels. However, this idea has two significant drawbacks.

First, decreasing the resolution of the images results in a loss of information, especially when aiming to achieve the small size of 11×36 for the projected lattice. This reduction in resolution would particularly impact the prediction results for images with shadows on the road or when distinguishing the road from side-walks, leading to poorer segmentation outcomes.
Second, our objective extends beyond road segmentation alone. We also aim to detect

and segment other objects present in the images, such as vehicles and pedestrians. Resizing the images is not a suitable solution for effectively segmenting small objects. Even if resizing is performed using down/up sampling techniques, it would require significant computational resources and increase computational time.
In summary, the use of resized images as an alternative to superpixels presents limitations in terms of information loss, particularly for challenging scenarios involving shadows and distinguishing road from side-walks. Additionally, resizing is not a suitable approach for segmenting small objects, and it introduces computational complexities. Therefore, leveraging superpixels remains a more advantageous choice for our multi-object segmentation objective.

a) regular SP b) irregular SP

Figure 3.12: Road segmentation prediction with SP-CNN for two samples based on: a) fixed-size superpixels (Patches), b) irregular superpixels

3.5.2 KITTI Evaluation on Birds Eye View Perspective

When evaluating the road segmentation in the birds-eye perspective on the KITTI benchmark, the images are first projected onto the ground plane using known camera geometry. This projection allows for a consistent representation of the scene in a regular grid format, enabling pixel-wise comparison (see section 3.4.1). The evaluation results in Table 3.3 provide insights into the performance of the segmentation algorithm across different categories (UM, UMM, UU, and URBAN) in the KITTI test dataset.URBAN category consists all three other subsets.

Comparing the results between table 3.1 and 3.3, we can observe, that the MaxF score in the pixel domain (90.08% from Table 3.1) is higher than the MaxF scores achieved by any of the individual benchmarks in the BEV perspective evaluation (ranging from 78.47% to 85.07% in Table 3.3). This indicates, that the segmentation algorithm performs better at the image perspective compared to the individual road categories evaluated in the BEV perspective.
The difference in performance can be attributed to the challenges introduced by the

BEV perspective evaluation. The error caused by inhomogeneous super-pixels at the road border, as mentioned before, is spread over a larger area due to terrain projection. This leads to a reduction in accuracy compared to the image perspective evaluation. To better understand the effect of terrain projection, Figure 3.13 provides visual examples of the mentioned projection effect on the results.

Benchmark	**M**axF	**A**P	**P**RE	**R**EC	**F**PR	**F**NR
UM_ROAD	81.60 %	69.62 %	78.13 %	85.40 %	10.89 %	14.60 %
UMM_ROAD	**85.07** %	**79.86** %	**85.97** %	**84.20** %	**15.11** %	**15.80** %
UU_ROAD	78.47 %	65.18 %	74.20 %	83.25 %	9.43 %	16.75%
URBAN_ROAD	82.36 %	72.31 %	80.48 %	84.33 %	11.27 %	15.67 %

Table 3.3: Evaluation Results on KITTI Test set

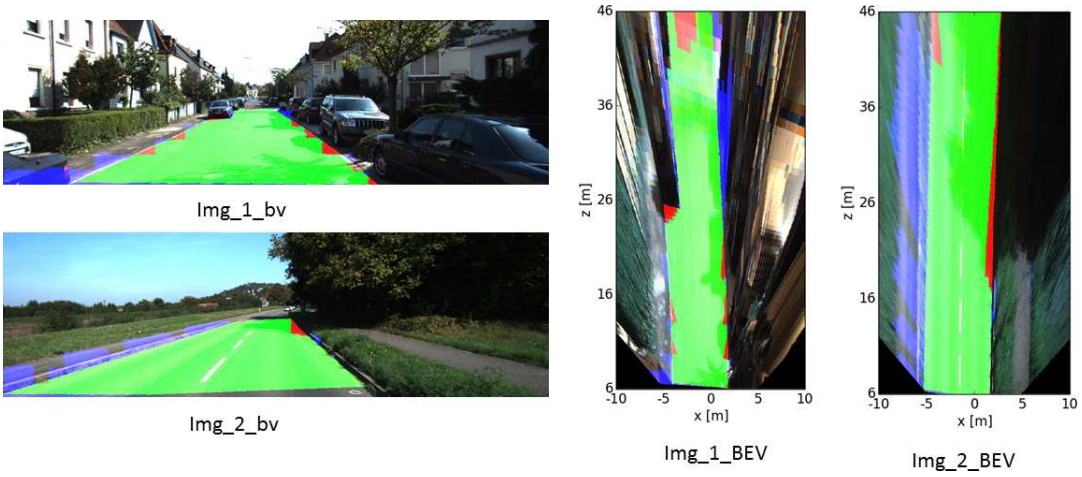


Figure 3.13: Road segmentation result from official KITTI test set in baseline and bird eye view perspectives. Here, red denotes false negatives, blue is false positives and green represents true positives.

3.5.3 Cityscape Evaluation

The evaluation results for road segmentation on the Cityscape validation dataset, categorized into pixel and superpixel domains, are presented in table 3.1. In order to facilitate a meaningful comparison between the results obtained from KITTI and

Cityscape datasets, we utilized the metric from KITTI for evaluating the results on the Cityscape dataset as well. However, during the evaluation on the test set, as elaborated in Section 3.4.1, we employed the intersection-over-union (IOU) metric [Cordts et al., 2016] to measure the accuracy of class predictions, alongside other state-of-the-art methods specifically designed for this dataset.

Evaluation on Validation Set

From table 3.1, in the superpixel domain, our approach achieves a slightly higher accuracy of 95.75%, compared to 94.10% in the pixel domain. This indicates that the superpixel-based segmentation approach yields slightly better overall accuracy in correctly classifying road segments within the Cityscape dataset.
Considering the F-measure, the superpixel domain outperforms the pixel domain with a score of 92.44% compared to 90.01%. This suggests that the superpixel-based approach achieves a better balance between precision and recall, indicating its effectiveness in accurately delineating road segments.
Analysing the precision values, the superpixel domain demonstrates a precision score of 89.08%, while the pixel domain achieves a slightly lower precision of 84.41%. This suggests that the superpixel-based approach is more successful in accurately identifying road pixels, achieving a higher percentage of correctly classified road pixels relative to the total number of pixels labelled as road.
In terms of recall, the pixel domain achieves a slightly higher recall (97.14%) than the superpixel domain (96.76%). However, both domains exhibit a strong ability to identify a high proportion of road pixels. The superpixel domain also demonstrates lower false positive rate (FPR) and false negative rate (FNR) compared to the pixel domain.
In summary, the superior accuracy of the superpixel-based approach in road segmentation on the Cityscape validation dataset can be attributed to the same reason as we discussed in the KITTI dataset. The key factor lies in the utilization of majority labels assigned to superpixels. It is important to acknowledge it, when superpixels exhibit variations in their composition, this assignment based on majority voting may introduce some inaccuracies at the pixel level. While this challenge exists, it does not significantly impact the overall results at the superpixel level. Despite these inherent challenges, the pixel-based approach still demonstrates notable performance in road segmentation.

Evaluation on Test Set

For the evaluation on the Cityscape test set, as elaborated in Section 3.4.1, we employed the intersection-over-union (IOU) metric [Cordts et al., 2016] to measure the accuracy of class predictions, alongside other state-of-the-art methods specifically designed for this dataset. Given that the publicly available Cityscape benchmark assesses 19 classes, it would be unfair to directly compare the average results of our approach (which focuses solely on the "road" class) with those of state-of-the-art methods. Hence, we conducted a comparative analysis by considering the per-class scores exclusively for the "road" class, examining how our approach fares against several

Method	Processor	Input Size	FLOPs(G)	Params(M)	road (IoU)	Runtime(S)	AI-Scores
DeepLab(LargeFOV-Strong)	Nvidia Titan X	512 × 1024	457.8	262.1	97.3%	4	24870
FCN	Nvidia Titan X	512 × 1024	136.2	512.5	97.4%	0.5	24870
CRF-RNN	Nvidia Titan X	512 × 1024	-	>513	96.3%	0.7	24870
Ours	**Intel(R) Core(TM) i7-4790K CPU @4GHz**	**32 × 64**	**0.253**	**0.3**	**90.08%**	**0.82**	**1400**

Table 3.4: Intersection of union(IoU) results for road class on Cityscapes testing set. All runtimes for each approach are obtained from Cityscape benchmark website [Cordts et al., 2016] and [Dong et al., 2020], which is evaluated with the same hardware. DeepLab LargeFOV-Strong [Chen et al., 2016], FCN [Long et al., 2015], CRF-RNN [Zheng et al., 2015]. AI-Scores:https://ai-benchmark.com/ranking_deeplearning.html

recent approaches on the Cityscape test set. The summarized results are presented in Table 3.4.
Furthermore, in addition to comparing the accuracy of our method with other state-of-the-art methods on the Cityscape test set, we also conducted a comprehensive analysis considering various factors such as the input size, number of FLOPs (floating-point operations per second), number of parameters (Params), runtimes, and AI-scores. The "AI-score" is a metric used to assess the performance of various hardware configurations, when running deep learning workloads. The higher the AI-score, the better the hardware configuration is considered to be in terms of its ability to handle deep learning tasks efficiently. By considering these multiple evaluation criteria, we aim to provide a comprehensive assessment of our approach in comparison to other methods, taking into account both accuracy and computational efficiency. These aspects will be discussed in detail in section 3.5.4.

From Table 3.4, although our approach achieved a lower accuracy compared to some state-of-the-art methods, it is important to highlight that our method offers notable advantages in terms of computational efficiency. Despite the slightly lower accuracy, our approach demonstrates comparable performance, while requiring minimal computational time.
Figure 3.14 showcases a selection of sample images from the Cityscape dataset. In the last row of the figure, both correct prediction labels and incorrect labels (shown in white) are displayed. Unlike many advanced semantic segmentation methods applied to urban scene datasets (as indicated in Table 3.4), which typically require downsampling the original high-resolution images from 1024×2048 to half the size (512×1024), our approach achieves impressive results with only occasional misprediction, primarily along the road boundaries, similar to our observations in the KITTI dataset. The ability of our approach to achieve such favourable results without the need for ex-

tensive downsampling highlights its effectiveness in accurately capturing the intricate details of the road segmentation task on the Cityscape dataset.

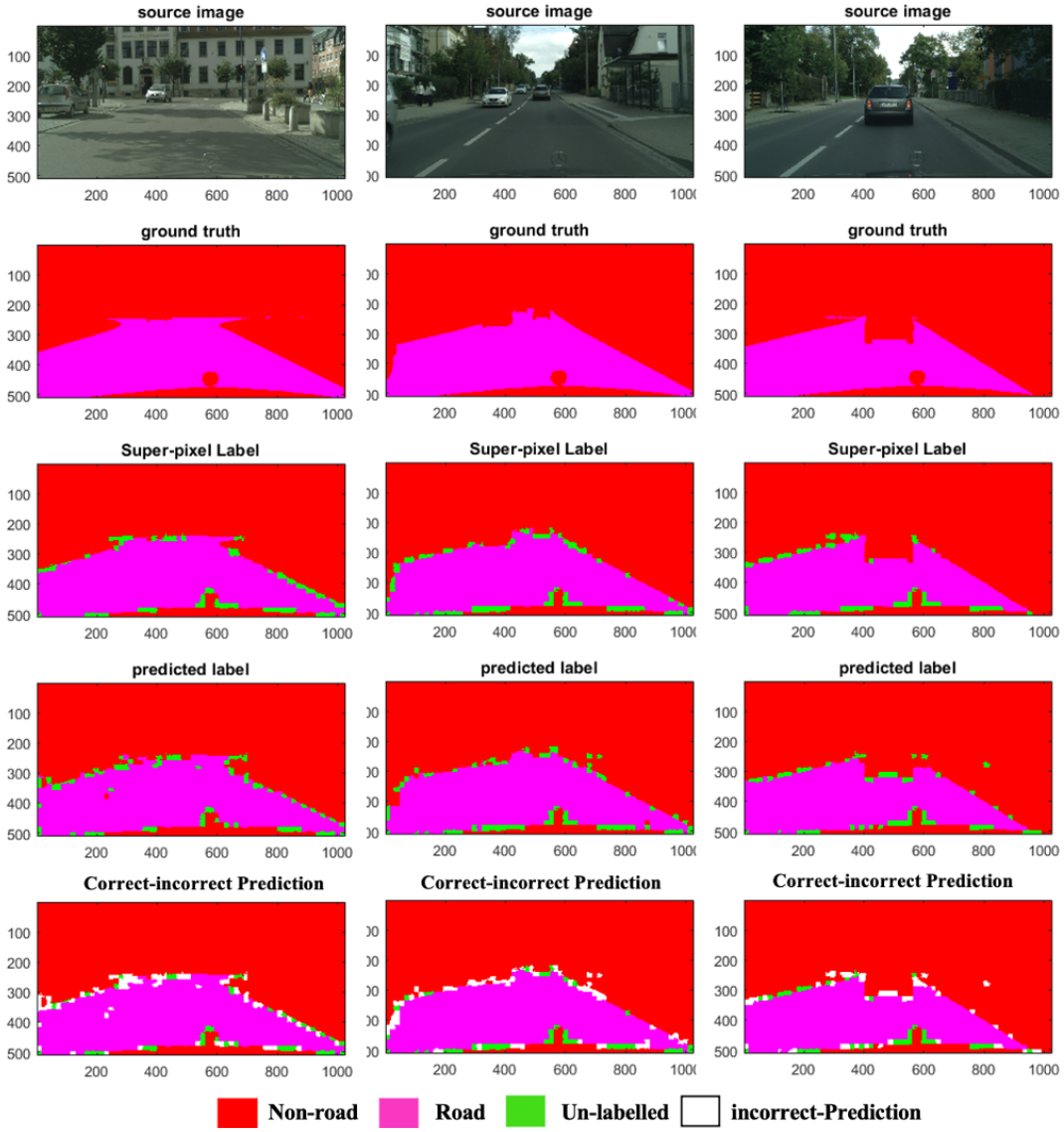


Figure 3.14: Road segmentation result from official Cityscape validation set. The first horizontal row shows input images. Ground-truth based on pixels and superpixels are shown in the second and third lines. The Fourth line shows the predicted results. The last row presents the correct and incorrect labelling prediction. The white colour specifies which parts are predicted wrong, which are mostly happened at along road boundary.

3.5.4 Run-time Analysis

Most of the state of the art methods in semantic segmentation are based on GPU and powerful hardware facilities, which limit their application to CPU based embedded systems. Our approach using superpixels and simple CNN network (see section 3.3.3) aimed at real-time performance, which reduces the computational complexity to ease the integration of proposed method into ADAS and autonomous driving systems. We evaluated the computational time for both KITTI and Cityscape datasets in both train and test time.

Training and Testing Run-time Comparison

During the experiments conducted on the KITTI road dataset, our approach not only achieved notable results in terms of accuracy, but also demonstrated remarkable efficiency. The training phase exhibited impressive speed, with a mere **4.3*s*** per epoch. It is important to note that we conducted a total of 200 epochs for all images, ensuring thorough model optimization and learning.
When it comes to testing, our approach showcased outstanding computational efficiency as well. The total run-time per image, including superpixel segmentation ($0.2s$), feature extraction (LBP $0.2s$ + position $0.002s$), and CNN learning ($0.009s$), amounted to an impressive **0.41*s***. These computations were carried out on a CPU, utilizing a Matlab implementation without parallel processing, as detailed in Section 3.5. For a comprehensive overview of the required running times for both the KITTI and Cityscape datasets, please refer to Table 3.5.

The swift training and testing durations of our approach demonstrate its efficiency in achieving accurate road segmentation results. By effectively utilizing computational resources and streamlining the processing steps, we have ensured that our approach delivers high-quality outcomes within a remarkably short time frame.

Dataset	**N**o_SP	**S**P	**F**E	**C**NN	**T**otal_Runtime
KITTI	396	0.2s	0.2s	0.01s	**0.41s**
CityScape	2048	0.3s	0.6s	0.02s	**0.82s**

Table 3.5: **CPU**-based Computational Run-Time in seconds per frame for both KITTI and City-scape image resolution: 1) number of superpixels(No_SP) 2) superpixel segmentation (SP), 3) Feature Extraction(FE), and 4) Network (CNN)

Comparison with State-of-the-Art Methods on KITTI Dataset

For the preprocessing steps (superpixel segmentation and feature extraction), we used non-optimized Matlab implementations for experimental reasons. However, runtime optimized versions for superpixel segmentation in CPU processor [Neubert, 2015] can reach a performance of $0.008s$ for an image segmentation into 400 superpixels. Further, an optimized version for LBP feature extraction [López et al., 2014] can reach $0.001s$. Using both bears the potential to squeeze the total runtime to $0.019s$, which is very fast for semantic segmentation without using GPU and enabling for embedding in the real time systems. In table 3.6, we compared our approach with some of the state of the art methods in semantic segmentation in both accuracy and computational time on KITTI dataset. All results provided for *URBAN road* category from test set.

Method	**P**rocessor	**M**axF	**A**P	**R**untime(s)	**A**I-Scores
LODNN	NVIDIA GTX980Ti GPU, 6GB memory	94.07 %	92.03%	0.018	16038
UP_CONV_POLY	NVIDIA Titan X GPU.	93.83 %	90.47	0.083	24870
DDN	NVIDIA GTX980Ti GPU, 6GB memory	93.43 %	89.67 %	2	16038
Ours (un-Optimized runtime)	Intel(R) Core(TM) i7-4790K CPU @4GHz	82.36 %	72.31 %	**0.41**	1400
Ours (Optimized runtime)	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	82.36 %	72.31 %	**0.019**	**1400**

Table 3.6: Comparative analysis of road segmentation performance on the KITTI UMM test set, featuring our SP-CNN method against some of the leading state-of-the-art methods. Only results of published methods are reported. LODNN: [Caltagirone et al., 2017], UP_CONV_POLY [Oliveira et al., 2016], DDN [Mohan, 2014].
AI-Scores:https://ai-benchmark.com/ranking_deeplearning.html

Despite the impressive run-time of **0.018*s*** achieved by the state-of-the-art method [Caltagirone et al., 2017], it is important to note that their implementation utilizes a high-performance NVIDIA GTX980Ti **GPU** with 6GB memory. This GPU-based processing significantly boosts their computational speed, making it approximately 11 times faster than our **CPU**-based method, which operates on a 4-core CPU. Thus, the disparity in run-time is expected given the hardware differences between the two approaches.
It is worth mentioning that the lower evaluated **MaxF** of our proposed approach, as discussed in the evaluation on Birds Eye Perspective (Section 3.5.2), is partly attributed to the presence of inhomogeneous superpixels at the road border. This

error becomes more pronounced due to terrain projection, resulting in a spread of inaccuracies over a larger area. However, it is worth noting that this limitation can be mitigated by implementing a post-processing step, potentially improving the **MaxF** to 96% for the KITTI dataset and 94% for the Cityscape dataset. Details regarding this post-processing step will be discussed in the subsequent chapter.
By acknowledging the hardware differences between our method and the state-of-the-art approach and addressing the challenges posed by inhomogeneous superpixels, we can better understand the observed discrepancies in run-time and evaluation scores.

Comparison with State-of-the-Art Methods on Cityscape Dataset

Finally, we compared the proposed method with several state-of-the-art methods on cityscape [Cordts et al., 2016] *test* dataset as a high resolution dataset. In Table. 3.4, we already reported the accuracy results (IoU) for class road. In addition, the corresponding running time as well as the status of the number of Parameters and FLOPs obtained for all the competing methods is presented here. Note that all the competing methods values are obtained from [Dong et al., 2020], where evaluated by using a single *NVIDIA TITAN X* card.

Given that Cityscape images have a high resolution of 2048×1024, most networks struggle to fit them into GPU memory, even powerful ones. To address this, the authors in [Dong et al., 2020] trained their models on smaller crops of full-resolution images and processed all images at half-resolution during testing. However, due to the superpixel-based nature of our approach, we are able to work with the native resolution in both the training and testing phases without losing any information.
Table 3.4 showcases the significant improvements in speed and computational efficiency achieved by our proposed method, while maintaining an acceptable level of accuracy and utilizing cost-effective resources. Specifically, our method achieves an impressive **90.08%** Intersection over Union (IoU) for road segmentation at a speed of **0.82s**, requiring only **0.253G** FLOPs and **0.3M** parameters. It is worth mentioning, that while FCN [Long et al., 2015] achieves the best performance with a runtime of $0.5s$ and 97.5% road-class IoU, they rely on a powerful GPU that is **18** times faster than our CPU and operates on half-resolution images. Furthermore, our proposed method utilizes **538** times fewer FLOPs and **1708** times fewer parameters compared to FCN for a similar computational time. It is important to mention, that the runtime results obtained from our approach using an unoptimized MATLAB implementation.

However, an optimized version of superpixel creation for high-resolution images (from 1339×891 pixels to 35042×336 pixels) with 1000 superpixels on an *Intel Core i5-6200U* processor can achieve runtimes ranging from **5** to **21** ms [Draegert, 2017]. Therefore, the runtime for Cityscape resolution would be approximately $\mathbf{8}ms$.

In summary, our approach demonstrates exceptional computational efficiency and advantages compared to other methods, even considering the limitations of our MATLAB implementation. By leveraging superpixels, we effectively handle high-resolution

images while achieving competitive results. This makes our approach highly suitable for real-time systems, striking a favorable balance between accuracy and computational speed. Additionally, our method is compatible with inexpensive hardware resources, enabling fast processing times during both training and testing phases.

3.6 Conclusion

We have introduced a superpixel-based convolutional neural network approach for road segmentation, with the primary objective of reducing resource utilization and runtime, while maintaining a high level of accuracy. Our aim is to enable the deployment of our approach on embedded devices, particularly in applications such as ADAS (Advanced Driver Assistance Systems).
The fundamental concept behind our approach is to leverage superpixel units instead of individual pixels as input for the CNN. By projecting a superpixel segmentation onto a regular lattice structure, we preserve the topology and enable efficient convolution operations. This allows us to extract object characteristics from nearly homogeneous and irregular-shaped superpixel units.
Extensive experiments were conducted to evaluate our approach under various sources of uncertainty. The results demonstrate promising levels of accuracy and efficiency in segmentation tasks, while significantly reducing processing time during runtime. In fact, our approach achieves state-of-the-art processing time scores for semantic road segmentation.
By adopting this superpixel-based strategy, we have effectively addressed the challenges of resource consumption and runtime in pixel-wise classification. Our method opens up possibilities for real-time deployment on embedded devices, making it a valuable contribution to the field of road segmentation.
Furthermore, the proposed superpixel-based mechanism is not limited to road segmentation alone. It can also be extended to facilitate semantic understanding of various objects in different domains. By leveraging the advantages of superpixel units, our approach has the potential to enhance the accuracy and efficiency of semantic segmentation tasks beyond road scenes. This flexibility and versatility further highlight the significance and applicability of our method in the broader context of object segmentation and understanding.
In the next chapter, we will delve into areas of improvement and expansion that can be explored based on our CPU-based superpixel CNN approach for semantic segmentation. While our approach has significantly reduced computational time, it is important to address the existing accuracy gap compared to GPU-based state-of-the-art methods. We will discuss strategies to mitigate the limitations arising from non-revisable decisions made during superpixel segmentation, focusing on enhancing the segmentation performance at road boundaries. These advancements will contribute to more precise and reliable road segmentation, improving the overall performance of our approach.

4 Efficient fine-grained road segmentation using superpixel-based CNN and CRF models

In this chapter, we continue the study of an efficient solution for semantic segmentation task applicable for real time systems.

While Convolutional Neural Networks (CNN) have shown great promise for semantic segmentation, their high computational requirements remain a significant obstacle. In a previous chapter, we introduced a novel approach that leverages the advantages of CNNs, while drastically reducing computational effort. We applied our proposed method for the task of road segmentation as a rudimentary problem in advance driver assistance systems (ADAS) to provide a safe and comfortable driving. By using irregular superpixels as the input to the CNN instead of the pixel grid, we achieved a notable reduction in runtime. However, the lower resolution of the superpixel domain resulted in decreased accuracy compared to state-of-the-art methods with higher computational costs.
This chapter presents an enhancement to the proposed semantic segmentation model by incorporating a Conditional Random Field (CRF) [Lafferty et al., 2001] to improve segmentation accuracy through the inclusion of spatial context. We consider semantic image segmentation again in the street and urban scene context as the specified task and investigate, how to improve the segmentation accuracy by incorporating spatial context. Our refinement procedure focuses on the superpixels that intersect with the predicted road boundary, ensuring minimal additional computational effort. Reducing the input to the superpixel domain allows the CNN's structure to stay small and efficient to compute, while keeping the advantage of convolutional layers. The integration of CRF compensates for the trade-off between accuracy and computational cost. The proposed system achieves performance comparable to top-performing algorithms on publicly available road benchmarks, and its fast inference makes it particularly well-suited for real-time applications.
This chapter is structured as follows. In section 4.1, we provide a clear understanding of the motivation and goals driving our work in this chapter. To offer the reader a comprehensive view, section 4.2 offers an up-to-date overview of the refinement procedures combined with the CNN-based model for semantic segmentation tasks. In section 4.3, we delve into the implications of Conditional Random Fields (CRF) as a type of probabilistic graphical model. CRF is utilized as a post-processing step on the results obtained from the previous chapter. We explain the fundamental principles of CRF and provide additional background information relevant to our work. Section 4.4 briefly reviews the scheme of our proposed method from the previous chapter, which

serves as a foundation for our current work in obtaining fine-grained segmentation results. Furthermore, we present an overview of our proposed technique to enhance the segmentation results. In section 4.5, we analyse three different CRF techniques that strike a balance between efficiency and accuracy. These techniques are evaluated and discussed in terms of their performance. To evaluate our refinements, section 4.6 assesses the three strategies from two perspectives: image perspective 4.6.1 and birds-eye view perspective 4.6.2. The evaluation is based on accuracy and time efficiency. Sections 4.7 summarize the chapter, discussing its limitations and drawing conclusions.

The findings presented in this chapter have been partially published in previous work [Zohourian et al., 2018c] or [Zohourian et al., 2022].

4.1 Introduction

Significant advancements in deep convolutional neural network (CNN) models, such as those introduced by [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2016], have greatly improved pixel-wise semantic segmentation tasks by extracting rich hierarchical features [Long et al., 2015, Zheng et al., 2015, Lin et al., 2016, Chen et al., 2014, Liu et al., 2015]. However, achieving fast and accurate pixel label estimation, that is compatible with real-time applications remains a challenging task. While recent approaches have increased accuracy by constructing deeper networks with numerous layers [Simonyan and Zisserman, 2014, He et al., 2016], these models often face limitations in computational power and memory. Additionally, many advanced CNN models, particularly those used for semantic image segmentation, require high-end GPUs for real-time inference, which may not be readily available for certain applications like mobile phones or robotic platforms.
In the previous chapter, we addressed these challenges by introducing a memory-efficient and fast semantic segmentation model, known as SP-CNN, which operates on CPUs at acceptable frame rates and consumes minimal computational resources. To demonstrate the applicability of this CPU segmentation model, we applied it to the task of urban scene segmentation, which is crucial for self-driving intelligent vehicles.
The proposed method distinguishes itself from conventional semantic segmentation approaches in two key aspects. First, it utilizes superpixel units rather than pixels as the input data model for the CNN network. Second, it adopts a simplified CNN layering structure. While most segmentation approaches achieve improvements in fully-supervised semantic segmentation through deeper network layering [Simonyan and Zisserman, 2014, He et al., 2016], which can be computationally expensive and require powerful GPUs for real-time embedding, our approach with a simplified network structure and larger input units (superpixels) reduces parameter count, enhances training and runtime efficiency, and enables execution on CPUs.
Despite achieving significantly faster speeds and requiring less memory, the segmentation accuracy of this method is relatively low, comparable to current state-of-the-art methods in semantic segmentation. Accurate road segmentation is challenging due

to variations in illumination, appearance, and occlusions. Furthermore, CNN-based methods predict labels independently for each local neighbourhood without incorporating information from surrounding labels, leading to coarse outputs. For example, parts of the road near the roadside area may be incorrectly predicted as a side-walk due to the lack of global context information. Additionally, the lower resolution of the superpixel domain naturally limits the ability to achieve fine segmentation results, particularly for small objects or heterogeneous superpixels.
In this chapter, we aim to enhance the efficiency of road segmentation by leveraging a conditional random field (CRF) technique [Lafferty et al., 2001] in conjunction with the previously proposed superpixel CNN-based segmentation approach. This further improves the efficiency of the model for CPUs and low-end GPUs (see Figure 4.1).Our segmentation algorithm follows several steps, which we summarize briefly here:

- Image segmentation into superpixels, where these coherent regions consist of pixels with similar image features
- Determination of image descriptors for the superpixels, consisting of multiple image features
- Assignment of superpixels to corresponding positions on a regular grid structure spanning the entire image
- Feeding the lattice and image descriptors into the convolutional network for superpixel classification based on semantic categories
- Development of an effective optimization strategy for increasing road segmentation precision using the CRF technique.

The current work comprises two aspects for coupling local and global evidences. To get more accurate and smooth segmentation results, we applied a Conditional Random Field (CRF)-based method as post prepossessing step. We combine the local image classification information extracted from CNN part with global information of neighbouring pixels to decide accurate pixel label. This idea follows largely previous work by fully connected pairwise CRFs [Krähenbühl and Koltun, 2011] as a post-processing step [Chen et al., 2014, Chen et al., 2016], integrating CRFs into the network by approximating its mean-field inference steps [Zheng et al., 2015, Liu et al., 2015, Lin et al., 2016] to enable end-to-end training. Experiments show that without using of global classification, the segmentation performs poorly, especially in inhomogeneous superpixels. However, with a hypothetical solution, the segmentation outperforms the CNN results and achieves comparable segmentation accuracy with other state of the arts methods. We demonstrate the power of the proposed architecture on public KITTI data set for road segmentation task.

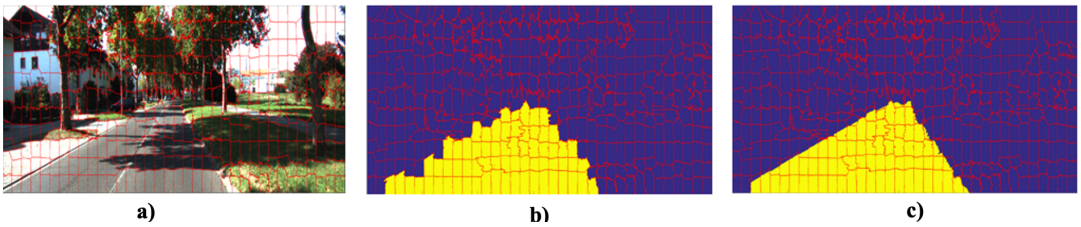


Figure 4.1: a) Original image overlay with superpixel segmentation, b) CNN segmentation result based on superpixel, c) CRF segmentation result.

4.2 Related Works

Motivated by the remarkable success of deep learning, convolutional neural network (CNN)-based classifiers have been adapted for the task of semantic segmentation. Semantic segmentation provides valuable information, including object boundaries and shapes, which are crucial for effective path planning and perception of the drivable road environment in self-driving intelligent vehicle systems. Earlier approaches primarily relied on a multi-patch approach [Giusti et al., 2013, Li et al., 2014]. However, more recently, state-of-the-art semantic segmentation systems have embraced the use of fully convolutional networks (FCNs), initially introduced in [Long et al., 2015]. These FCNs employ a skip architecture to integrate heat maps from different resolutions and replace the last few fully connected layers of a conventional CNN with convolutional layers, enabling efficient end-to-end learning with variable input sizes. Various enhancements and deeper variants of FCNs have been proposed in [Badrinarayanan et al., 2015, Noh et al., 2015].
However, CNN-based segmentation methods encounter certain issues that can impact the accuracy of semantic segmentation, leading to imprecise region segmentation and discontinuous edges. One key issue is that CNN-based methods independently predict labels for each local neighbourhood without considering information from surrounding labels. This is despite the fact that labels produced by the CNN at neighbouring pixels are often correlated due to the overlap of their receptive fields. Unfortunately, these dependencies are not explicitly modelled, resulting in coarse outputs when convolutional filters with large receptive fields are employed. Furthermore, consecutive pooling operations during the encoding stage of CNNs can diminish the likelihood of achieving fine segmentation results, particularly for small objects such as pedestrians or traffic lights. This loss of detailed information arises from the pooling operations. To address these challenges and enhance the accuracy of pixel labelling, CNNs are often combined with probabilistic graphical models like Conditional Random Fields (CRFs) [Lafferty et al., 2001]. CRFs are utilized to model various dependencies among pixel labels and refine weak and coarse segmentation outputs in a post-processing step [Zheng et al., 2015, Chen et al., 2014, Chen et al., 2016]. [Farabet et al., 2013] utilized a superpixel-based CRF model as a post-processing step on a CNN-based multi-scale feature extractor to achieve the final segmentation. Another method, DeepLab-CRF [Chen et al., 2014], employed a dense CRF approach [Krähenbühl and Koltun, 2011]

as a post-processing step on a trained Fully Convolutional Network (FCN). [Zheng et al., 2015] introduced an end-to-end CNN-CRF model, integrating the CRF directly into the CNN architecture and jointly learning the CRF parameters and CNN parameters from training data. They also employed a mean-field inference technique, implemented as a Recurrent Neural Network (RNN), to optimize the dense CRFs. This approach was later extended to handle higher-order CRF terms [Arnab et al., 2015]. Extensions of the RNN-CNN approach, such as the bilateral neural network for high-dimensional sparse data [Jampani et al., 2016], have also been proposed.
Our proposed model differs from these state-of-the-art approaches in the way we combine CRFs and CNNs. We utilize pixel-level CRFs as a proposal mechanism for a CNN-based re-ranking system, focusing only on a small portion of pixels surrounding the road border. In contrast to [Farabet et al., 2013], who used superpixels as nodes for a local pairwise CRF, we consider each pixel belonging to the superpixels that intersect with the road contour as a CRF node. This approach exploits long-range dependencies, while significantly reducing the computational cost associated with performing pixel-level CRF inference on the entire image resolution.

4.3 CRF Background and Notations

In this section, we will delve into the fundamental concept of Conditional Random Fields (CRF) [Lafferty et al., 2001], as a class of statistical modelling methods.

4.3.1 CRF Basics and Comparison with MRFs

CRFs are undirected probabilistic graphical models widely employed for capturing spatial regularities in images. While a classifier predicts labels for individual samples without considering neighbouring samples, CRFs have the ability to incorporate contextual information. This is achieved by modelling the predictions as a graphical model, that represents the dependencies between them. CRFs serve as a common type of machine learning approach used across a wide range of applications, from mitigating low-level noise to performing high-level scene interpretation. Their objective is to assign appropriate class labels to each pixel, considering the relationships and context within the image.
In image segmentation, early works primarily centred around the use of a probabilistic generative framework known as Markov Random Fields (MRF) [Bishop and Nasrabadi, 2006, 383-393]. This framework models the joint probability of an image X and its corresponding labels Y. MRF-based approaches consider the image and its labels as a graphical model, where nodes represent image pixels or regions, and edges represent the relationships and dependencies between these nodes. The joint probability distribution is defined based on the local interactions between neighbouring pixels or regions, capturing the spatial coherence and smoothness properties of the image. The goal is to determine the most likely set of labels, that accurately represent the underlying structure and boundaries within the image.

4.3.2 CRF Structure and Advantages

While MRF offers advantages, it also comes with significant limitations, particularly when dealing with a large dataset or complex dependencies among features. In such cases, constructing a probability distribution over all variables with a large number of components in X and Y becomes challenging. Additionally, modelling dependencies between inputs can lead to complex and intractable models, while ignoring these dependencies can result in performance degradation.
A promising solution to address this challenge is the use of Conditional Random Field (CRF) [Lafferty et al., 2001]. CRF takes a discriminative approach, combining the strengths of discriminative classification and graphical modelling. Specifically, CRF models focus on the conditional distribution $P(Y|X)$, where Y represents the multivariate outputs $Y = (y_1, y_2, ..., y_n)$, with each y_i corresponding to the class label assigned to a specific pixel in the context of image segmentation. On the other hand, $X = (x_1, x_2, ..., x_n)$ represents a set of random variables, that denote observed evidence for pixels, such as intensity or colour values. The domain of Y is a set of labels $L = \{l_1, ..., l_C\}$ in C classes. For a binary class segmentation, $C = 2$ and $L = \{0, 1\}$. To compactly capture the dependencies between variables, CRF utilizes an associated graphical structure $G = (V, E)$. This undirected graph consists of nodes (or vertices) V, which represent individual pixels in the image, and edges E, that denote pairwise

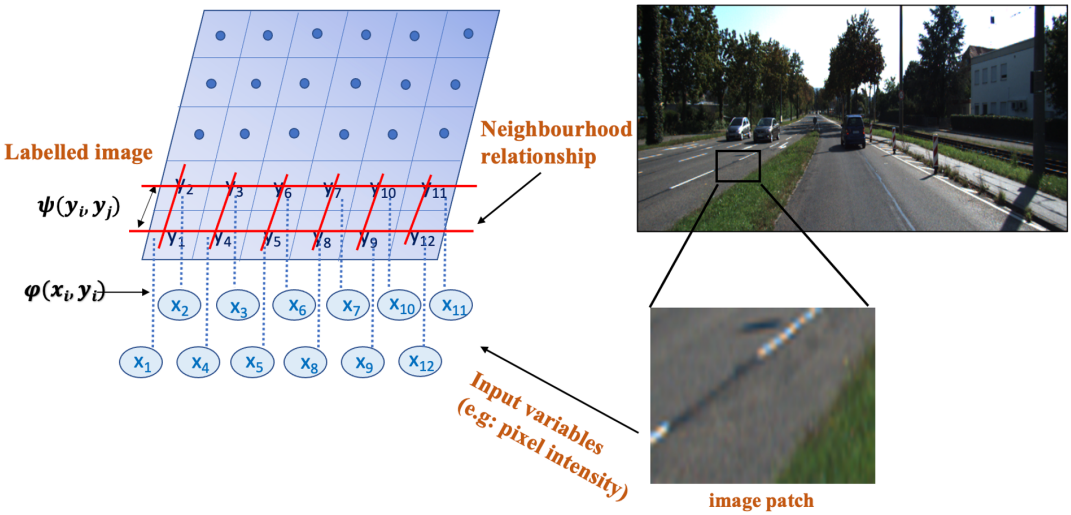


Figure 4.2: Illustration of a Conditional Random Field (CRF) Model for Pixel-wise Image Labelling. In this diagram, each pixel is represented by y_i, indicating its label, and x_i, denoting its characteristic features such as colour, intensity, or texture. The inter-pixel relationships are depicted by red lines, illustrating the connectivity among neighbouring pixels and implying adherence to the Markov property. Blue lines illustrate the association between each pixel and its corresponding label. This representation is based on concepts from [Yedidia et al., 2003, Freemann, 2012].

relationships between neighbouring pixels. These edges allow the CRF to effectively model dependencies between adjacent labels. In Figure 4.2, a comprehensive visual representation of the concept is shown. In the graph, the total number of nodes is denoted as $n = |V|$, with each pixel associated with an observed image feature x_i and an output variable y_i (segmentation label). The output variables are conditioned on the input variable x_i, following the Markov property with respect to the graph structure: $P(y_i|x_i, \{y_j : i \neq j\}) = P(y_i|x_i, \{y_j : i \sim j\})$. This equation says, that the probability of y_i given x_i and the labels of all other nodes (excluding i) is equal to the probability of y_i given x_i and the labels of its neighbouring nodes j, where $j \sim i$ indicates that nodes i and j are neighbours in the graph G. Typically, neighbouring pixels are those that are spatially close to each other, like immediate horizontal and vertical neighbours. The definition of this neighbour system, including the choice of adjacency or nearest neighbour relations, is crucial in determining the proximity of the contextual constraint.
This approach offers several advantages, such as simplifying the model structure compared to joint models and significantly enhancing computational efficiency. Unlike the generative nature of MRF, CRF is a discriminative graphical model, that primarily emphasizes the posterior distribution of observations. This enables CRF to directly predict the unobserved variables based on the observed ones. By focusing on the conditional distribution, CRF effectively leverages the available information to make accurate predictions, making it well-suited for various tasks, including image segmentation.

4.3.3 Parameter Estimation and Energy Minimization in CRFs

The crucial step for effectively utilizing CRFs is the estimation of model parameters to accurately capture the underlying relationships between observed evidence and class labels. Expectation Maximization (EM) [Bishop and Nasrabadi, 2006, Moon, 1996],a cornerstone in probabilistic model parameter estimation, is an iterative algorithm used to estimate the parameters of a probabilistic model in the training phase, when there are hidden or unobserved variables. During the Expectation step (E-step), the algorithm estimates the posterior probabilities of the hidden variables, given the observed data. In the Maximization step (M-step), the algorithm maximizes the expected log-likelihood of the data respect to the model parameters. By iteratively alternating between these two steps, EM optimizes the CRF parameters, which is important for accurate and effective CRF modelling.

After estimating the model parameters using EM or other optimization methods, the core principle of CRFs revolves around energy minimization. The primary objective is to establish an energy function, that determines the most likely assignment of output variables (pixel labels) based on the input data. In the realm of energy-based segmentation, this predication is mathematically formalized through the Gibbs distri-

bution [Krähenbühl and Koltun, 2011], a framework that articulates the probability of a label configuration in light of input data and model parameters:

$$P(Y|X) = \frac{1}{Z(X)} \exp(-E(Y|X)) \tag{4.1}$$

Here, $E(Y|X)$ represents the energy of a labelling configuration Y, given the observed features X. This formulation indicates, that the probability of a labelling configuration Y is inversely related to its Gibbs energy, which means lower energy implies higher probability and vice versa. This energy is defined as:

$$E(Y|X) = \sum_{c \in C_G} \phi_c(Y_c|X) \tag{4.2}$$

In the above equation C_G is the set of cliques of the graph $G = (V, E)$ on Y. Y_c is the set of labels for the pixels within clique c. ϕ_c is the clique potential function, also known as the compatibility function, which measures the agreement of the label configuration Y_c with the observed evidence X within the clique c. $Z(X)$ is a normalization factor, ensures that the probabilities sum to one. The Gibbs energy of a labelling measures, how compatible a possible labelling of the image is with the observed evidence and with itself.

Transitioning from clique-based formulations to a more granular perspective, CRFs can be expounded in terms of unary and pairwise potentials, particularly germane to image segmentation. The unary potential quantifies the likelihood of each pixel belonging to a certain class, informed by observed data, while the pairwise potential, or smoothness term, fosters spatial consistency among adjacent pixels. This dichotomy not only encapsulates the relationships between individual labels and local features, but also between labels of neighbouring pixels, thereby embodying the spatial coherence intrinsic to image data.
Accordingly, the energy function for a labelling Y, integrating both unary and pairwise potentials, can be succinctly expressed as [Krähenbühl and Koltun, 2011]:

$$E(Y|X) = \alpha \sum_i \varphi(x_i, y_i) + \beta \sum_{ij} \psi(y_i, y_j) \tag{4.3}$$

where α and β are some positive constant weights. $\varphi(x_i, y_i)$ represents the unary potential for pixel i, that is getting a particular label y_i. This value can be computed for instance from CNN and corresponding annotated label. Function φ pays a penalty for disregarding the predicted label from annotation. The second term is the pairwise potential or the smoothness term, representing the joint Gibbs distribution on the label field and satisfying the Markov property. It encodes the neighbouring

information between two labels of neighbouring pixels i and j and ψ_{ij} computes the difference between a pixel label from its neighbour.

4.3.4 Inference Methods in CRFs

We want to find the optimal labelling Y^*, that maximizes the posterior probability $P(Y|X)$ given the observed image X. MAP [Nowozin et al., 2011] estimation can be expressed as:

$$(MAP)Y^* = \arg\max_Y P(Y|X) \tag{4.4}$$

This is equivalent to finding the labelling that minimizes the Gibbs energy:

$$Y^* = \arg\min_Y E(Y|X) \tag{4.5}$$

where Y^* is the most probable pixel-level image labelling from a set of all possible hidden variables (labels) for the observable variable X.

The aim is to find the label configuration, that minimizes the overall energy, thereby yields the most probable segmentation. The inference process involves finding the best label configuration Y^*, that minimizes the energy function $E(Y|X)$ for a given input image X. Finding exact inference in CRFs is intractable and computationally expensive due to the complex dependencies and the large number of possible label configurations. Therefore, approximate inference methods are often employed, such as Iterated Conditional Mode (ICM) [Besag, 1986], Belief Propagation (BP) [Bishop and Nasrabadi, 2006, 393-418], mean field approximation [Tanaka, 1999], to efficiently estimate the most likely assignment of class labels. The goal is to assign labels to each pixel in the image, that maximize the posterior probability $P(Y|X)$. By combining estimation model parameter technique with approximate inference methods, the CRF model can efficiently and effectively estimate the most likely assignment of class labels to pixels.

We propose a pixel-wise image segmentation method based on statistical classification of Markov fields. We treat the image labels as hidden states and infer them using different algorithms that maximize the posterior probability. We assume the same data distribution for all models, but we vary the Markovian prior assumptions over the label random field. We use CNN to provide the unary potentials for all three conditional random field (CRF) models. The inference steps involve different types of interactions or message passing operations between the pixels, such as:

- **Nearest-neighbour interactions:** We sum up the spatially varying horizontal and vertical interactions between the pixels. This is also known as iterated conditional modes (ICM)(section 4.5.1).

- **Belief Propagation:** We propagate messages along the edges of the CRF graph, that represent the beliefs about the label of each pixel. We compute the marginal distributions of the label random field from these messages (section 4.5.2).
- **Mean Field updates:** We approximate the distribution of the label random field by a simpler distribution, that minimizes the Kullback-Leibler divergence [Kullback and Leibler, 1951, 79-86]. We update the parameters of this distribution iteratively until convergence (section 4.5.3).

We construct the CRF graph and potential functions from the image data, and apply them only to the image pixels that belong to the superpixels touching the predicted road boundary. We compare the performance of all three methods for classification and analyze the trade-off between computational efficiency and accuracy. Our method allows for robust and accurate statistical analysis on the road segmentation task.

4.4 Proposed Method: Enhanced Superpixel-CNN with CRF for Fine-Grained Semantic Segmentation

In our latest research, we have integrated Conditional Random Field (CRF) and Convolutional Neural Network (CNN) methodologies into a comprehensive framework tailored for fine-grained road segmentation in urban environments, showcased in Figure 4.3. This integration builds upon the groundwork laid out in Chapter 3, where our initial approach was introduced. In the upcoming section (4.4.1), we will briefly revisit this foundation. Understanding this background is crucial, as our current model, enriched with CRF techniques, is rooted in these foundational concepts. We further refined our approach by aligning superpixel contours with road boundaries using CRF. This targeted enhancement significantly increases accuracy with minimal computational overhead, as detailed in Section 4.4.2. Subsequent sections will delve into the specific techniques applied and the results obtained, providing a comprehensive insight into our refined road segmentation method.

4.4.1 Superpixel-based Convolutional Neural Network

In Chapter 4.4.1, we introduced a method that harmonizes the advantages of superpixels with convolutional capabilities. Our method leveraged irregular superpixel segmentation to enhance the efficiency and accuracy of convolutional neural networks (CNNs).

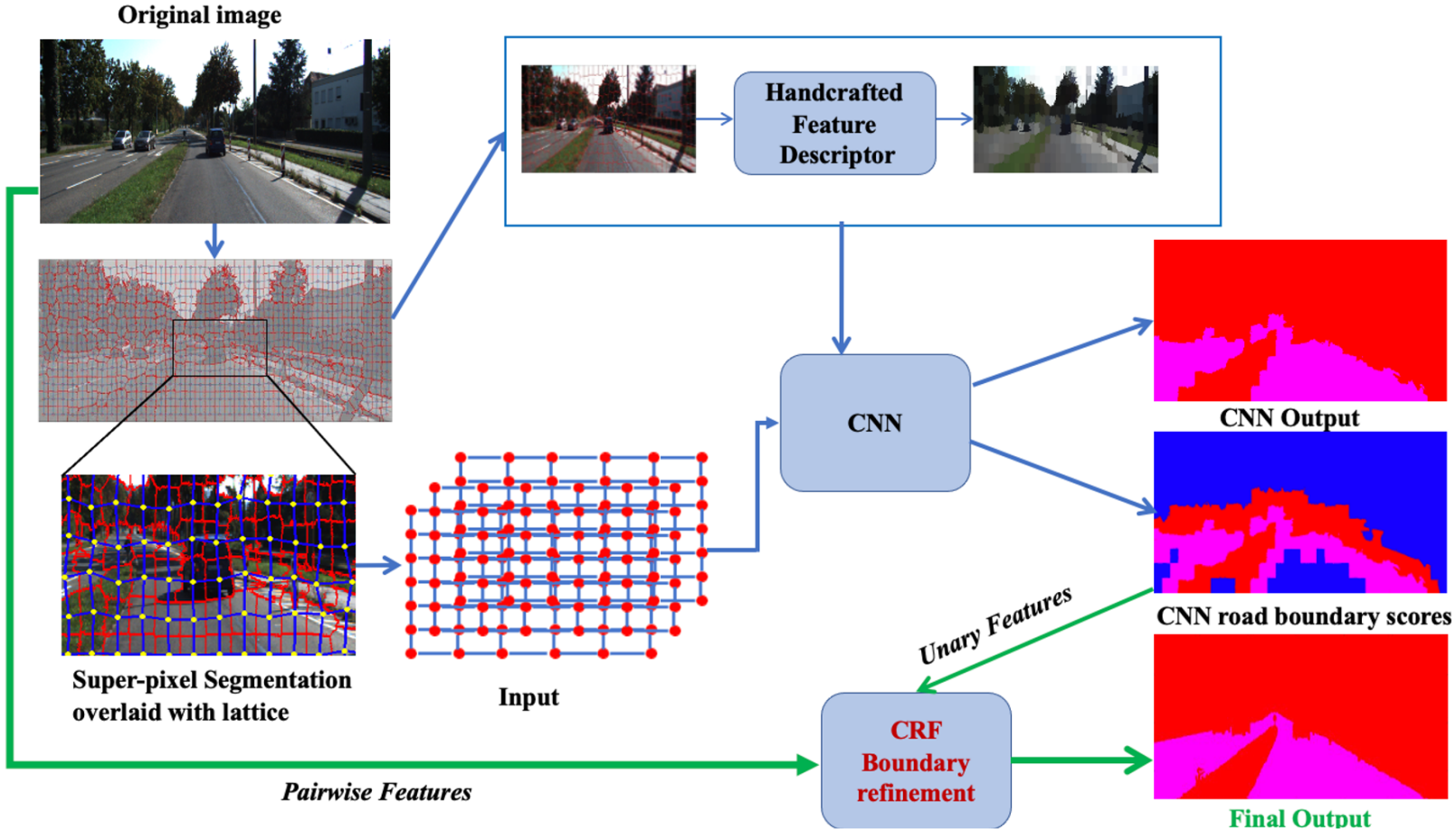


Figure 4.3: The Proposed refinement SP-CNN Architecture for road segmentation. Initially, superpixels are generated from the original input image. Subsequently, each superpixel is characterized using a high-dimensional local feature descriptor. These superpixel descriptors are then projected onto a lattice, and the resulting lattice projection along with descriptors are fed into a CNN for semantic category classification. Following CNN processing, a fully connected Conditional Random Field (CRF) technique is applied to the output, focusing on the road boundary to enhance the segmentation accuracy.

Input Data Model

Our process initiated with the segmentation of the image into coherent superpixels, utilizing an adopted version of the SLIC algorithm [Achanta et al.,]. SLIC employs k-means clustering in both spectral and spatial dimensions, leading to the growth and refinement of superpixels based on spectral-spatial distances. However, the inherent variability introduced by the "Enforce-Connectivity" procedure in SLIC makes the resulting superpixels unsuitable for direct CNN input. To address this, we made strategic modifications to ensure consistency and compatibility with CNNs. For more details you can see the section 3.3.1.
We preserved smaller regions and adjusted the segmentation to produce more consistent superpixels. In cases, where superpixels had multiple segments sharing the same label, we retained the larger one and merged the rest into the nearest superpixel. The selection of the nearest neighbourhood was based on Euclidean distance, measured between the centre of the sub-segment and the centre of each adjacent segment. Recognizing the necessity of a regular structure for convolution, a lattice was defined, centred on the rectangular structure from SLIC's initial iteration, projecting it onto corresponding irregular superpixels from the final step.
Additionally, we crafted a 69 dimensional image descriptor for each superpixel, incorporating various image features such as colour, position, and Local Binary Pattern (LBP) features. This unified data, coupled with image descriptors, formed the input for the CNN, facilitating pixel-wise classification with heightened accuracy and efficiency.

CNN Network Architecture

Our unique input data model, which benefits from the structure of superpixels and feature descriptors, allows us to use a simpler CNN architecture. This reduces computational time. The network consists of two convolutional layers, two channel-wise feature layers, and a drop-out layer. Each layer is followed by a non-linear activation function. The input is determined by the superpixel lattice, and the output classifies segments as either *road* or *un-road*. Further details on the network architecture can be found in Section 3.3.3.

4.4.2 Segmentation Refinement with CRF

Our superpixel-based convolutional network already has the capability to do the coarse segmentation by capturing the local properties such as object shape and contextual information. In addition to this local information, the global context information that captures semantic of spatial interdependencies is also required for an accurate image labelling. However, this information may not explicitly included into the CNNs. CNNs have their own shortcomings to directly model the interactions and correlation between the output variables to obtain this global information, which is important for a smooth semantic segmentation(see figure 4.4).

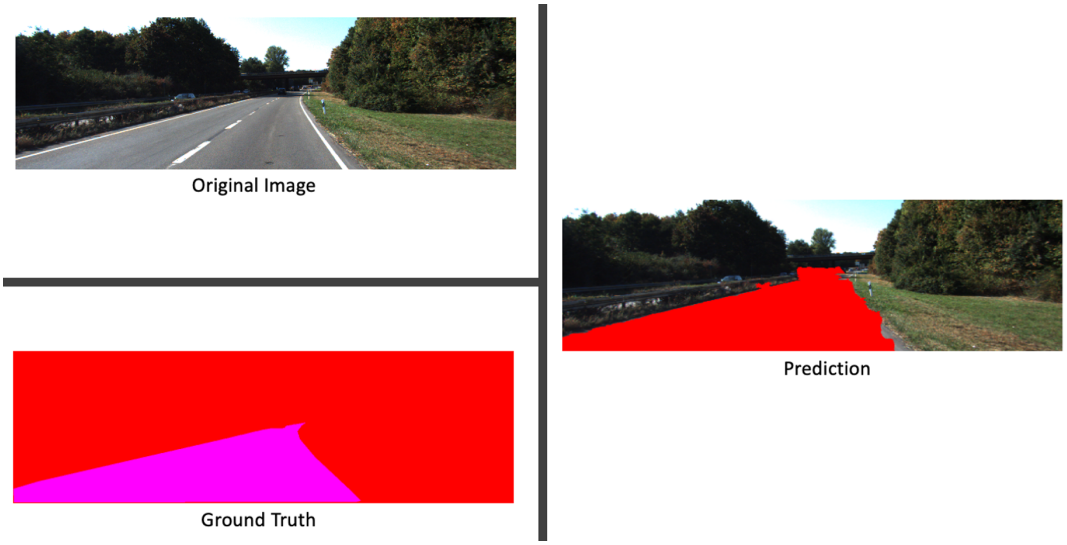


Figure 4.4: The prediction result from our proposed superpixel based CNN method from one sample of KITTI validation dataset.

To this end, Conditional Random Fields (CRF) [Lafferty et al., 2001] has been used to impose consistency and coherency between labels. As discussed in section 4.3, they can model global properties like object connectivity, geometric properties and spatial relationship between objects. By combining CNN and CRF models, we are able to fine-tune the CNN segmentation results for the task of road segmentation specially along the road border.

4.5 Customized CRF Optimization: Model Analysis for CNN Integration

In Computer Vision and image processing, many problems can be approached as energy minimization tasks, effectively modelled using Conditional Random Fields (CRFs). Particularly in image segmentation, CRFs aim to assign optimal labels to individual pixels. The precise determination of the "best" label for each pixel is pivotal and directly influences the definition of the energy function.
Building on our exploration of CRF fundamentals in the previous section, this chapter introduces our tailored adaptations of key inference methods: ICM, LBP, and Mean Field. These versions, enhanced by integrating the unary component from our CNN results, optimize energy minimization and enhance semantic segmentation. Following rigorous testing, the most effective technique will be selected from these customized approaches. Designed for both standard CPUs and resource-constrained GPUs, these CRF enhancements signify a notable advancement in image segmentation accuracy.

4.5.1 Iterated Conditional Modes (ICM)

Iterated Conditional Modes (ICM) [Besag, 1986] is an energy optimization method for MRF. It is an iterative algorithm, that employs a deterministic greedy strategy to find a local minimum by minimizing an energy function in equation 4.5. It employs knowledge of a neighbourhood system for optimal solution inference. It is initiated by an estimation of $P(Y|X)$ or a random selection and smooths out the initial segmentation by assigning the new label to each pixel so that it has the minimum energy given the neighbourhood relations. This process is repeated until convergence. The main steps of ICM can be written as follows:

1. Initialize the label configuration $Y^{(0)} = (y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)})$, where $Y^{(t)}$ is the label configuration for all pixels in the image at iteration t, n represents the number of pixels in the image, and each $y_i^{(0)}$ is the initial label for the i^{th} pixel. This initialization can be based on an estimation from $P(Y|X)$ or a random selection. Define the convergence criteria, including the maximum number of iterations (MaxItr), the energy threshold (T), and the minimum change in energy (ΔE) required to continue between iterations.

2. For each iteration t and each pixel i, compute the potential energy change $\Delta E_i^{(t)}$ for assigning any possible label l from the label set L. Calculate this energy change as:
$$\Delta E_i^{(t)}(l) = E(y_i^{(t)} = l|X) - E(y_i^{(t-1)}|X)$$
Here, $E(y_i = l|X)$ is the energy associated with assigning the new label l to pixel i at iteration t represents the energy of the current label configuration, $E(y_i^{(t-1)}|X)$ represents the energy of pixel i from the previous iteration $t-1$.

3. Update the label of pixel i to the one, that minimizes the energy,
$$y_i^{(t)} = \arg\min_l \Delta E_i^{(t)}(l)$$
This minimization leads to a lower energy state, i.e., $\min_l \Delta E_i^{(t)}(l) < 0$. If no label for any pixel leads to a lower energy state, conclude that a local minimum has been reached and terminate the algorithm.

4. Check for termination criteria. If the current iteration t reaches MaxItr, or if $\Delta E = |E^{(t)} - E^{(t-1)}| < T$, where $E^{(t)}$ is the total energy of the system at iteration t, and T is a small positive threshold value, stop the algorithm. Otherwise, increment t by 1 and repeat from step 2.

Although, ICM is an efficient method, which can be quickly completed after a few iterations, this method cannot guarantee a globally optimal solution, due to two main reasons: a) *Label initialization* and b) *Local convergence*. In comparison with other energy minimization methods such as Loopy Belief Propagation(LBP) [Murphy et al., 1999], ICM is greatly affected by the label initialization [Szeliski et al., 2006]. In case of a bad initialization, the error will spread out after iterations, whereas with a good

initialization, the result can be surprisingly satisfying. High-dimensional spaces with non-convex energies, such as pixel-wise segmentation in vision domain, are highly sensitive to the initial estimation. Therefore, choosing an appropriate initialization is really important in this domain. Moreover, due to their large number of local minima, ICM may result in a solution that is far from global optimum, which is not acceptable.
In the past, the initialization for ICM typically relied on the maximum likelihood method, yielding less promising classification accuracy. However, a notable enhancement emerged when ICM was initialized using Convolutional Neural Network (CNN) outputs. Within the ICM method, labels are assigned to pixels using an energy minimization approach, aiming to optimize the segmentation quality and accuracy. This method has become a pivotal aspect of our redefined ICM approach.

Algorithm (1) outlines the steps of our proposed ICM implementation. To estimate precise variance values for pixel intensity levels, we applied a Gaussian Mixture Model (GMM) [Williams and Rasmussen, 1996] tailored for our application. In the realm of mixture models, a latent variable is employed to denote a specific mixture component selected from a discrete set of K components. In our customized GMM, we integrated two distinct components, each corresponding to one of our classes, to model the pixel intensity distributions of road and non-road classes. Within our optimization framework, we introduced an auxiliary vector $\bar{k} = (k_1, ..., k_i, ..., k_n)$, where $k_i \in \{1, ..., K\}$ assigns a unique GMM component to each pixel. This assignment is based on whether the pixel is associated with the road ($y_i = 1$) or non-road ($y_i = 0$) class. Each component within a GMM is characterized by three fundamental parameters, the mean RGB value (μ), the covariance matrix (Σ), and the weighting coefficient (π) that enables their mixing. Through adjustment of the weighting coefficients, the model proficiently analyses pixel intensity distributions, facilitating precise classification into the individual classes. The energy function presented in equation 4.3 can be reformulated as follows [Rother et al., 2004]:

$$E'(Y, \bar{k}, \bar{\theta}, X) = \alpha \sum_i \varphi'(x_i, y_i, k_i, \bar{\theta}) + \beta \sum_{ij} \psi'(y_i, y_j) \tag{4.6}$$

In this equation, Y represents the set of labels for each pixel. X represents the features of each pixel. $\bar{\theta}$ represents the parameters of the GMM component, where $\bar{\theta} = \{\pi(y,k), \mu(y,k), \Sigma(y,k)\}$, $y \in \{0,1\}$, and $k \in \{1, \ldots, K\}$. α and β are penalize factors.

In GMMs, the goal is often to maximize the likelihood of the observed data given the model parameters, or equivalently, to minimize the negative log-likelihood. This approach transforms the product of probabilities into a sum of log probabilities, simplifying the computation, especially for optimization algorithms. Equation 4.7 likely

Adaptive ICM $(X, Y, T, MaxItr)$

Input : pixels contained in superpixels surrounding the road border (X), corresponding labels from CNN (Y), Potential threshold (T), maximum number of iterations $(MaxItr)$
Output: smooth road segmentation

1) Initialize Mixture Models parameters $\bar{\theta}$ from CNN outputs `// GMM Model`

while *not MaxItr* **do**

2) Assign GMM components

$$k_i^* = \arg \min_{k \in \{1,\dots,K\}} \varphi'(x_i, y_i, k, \bar{\theta})$$

3) Learn GMM parameters

$$\bar{\theta}^* = \arg \min_{\bar{\theta}} \sum_i \varphi'(x_i, y_i, k_i^*, \bar{\theta})$$

4) Calculate unary energy `// Unary Energy`

$$\varphi'(x_i, y_i, k_i, \bar{\theta}) = -\log \pi(y_i, k_i) + \frac{1}{2} \log |\Sigma(y_i, k_i)| + \frac{1}{2}(x_i - \mu(y_i, k_i))^T \Sigma(y_i, k_i)^{-1}(x_i - \mu(y_i, k_i))$$

5) Calculate pairwise energy `// Pairwise Energy`

$$\psi'(y_i, y_j) = \begin{cases} \omega, & \text{if } y_i \neq y_j \\ 0, & \text{if } y_i = y_j \end{cases}$$

6) Calculate the total Energy based on appropriate Penalize values

$$E'(Y, \bar{k}, \bar{\theta}, X) = \alpha \sum_i \varphi'(x_i, y_i, k_i, \bar{\theta}) + \beta \sum_{ij} \psi'(y_i, y_j)$$

7) Estimate optimal labels `// Total Energy`

$$\hat{Y} = \arg \min_Y E'(Y, \bar{k}, \bar{\theta}, X)$$

end

Algorithm 1: Our Developed ICM algorithm

represents the probability of a data point x_i under the GMM, expressed as a weighted sum of Gaussian distributions:

$$P(x_i|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \tag{4.7}$$

where, x_i represents a data point in the dataset, (could be a feature vector derived from a pixel). $\mathcal{N}$ denotes the Gaussian distribution function or the probability distribution function(PDF) and $\mathcal{N}(x_i|\mu_k, \Sigma_k)$ denotes the probability distribution function(PDF) for a normal random variable (Gaussian distribution) with mean μ_k and covariance Σ_k, and mixing coefficients π_k corresponding to individual mixture components k, so that:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \forall k \quad \pi_k \geqslant 0 \tag{4.8}$$

By definition, the probability density function (PDF) of observing a data point x_i is given by:

$$\mathcal{N}(x_i|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{4.9}$$

To optimize the parameters of the GMM, it is common to minimize the negative log-likelihood of data, leading to the equation:

$$-\log P(x_i|\theta) = -\log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)\right) \tag{4.10}$$

From the first part of the equation 4.6 for a single data-point x_i and by applying the negative logarithm of the likelihood, we can write:

$$\varphi'(x_i, y_i, k_i, \bar{\theta}) = -\log\ P(x_i|y_i, k_i, \bar{\theta}) - \log\ \pi(y_i, k_i) \tag{4.11}$$

Where $p(\cdot)$ is a Gaussian probability distribution, and $\pi(\cdot)$ are mixture weighting coefficients. Taking the negative logarithm of the likelihood aims to minimize the negative log-likelihood, equivalent to maximizing the likelihood, ensuring numerical stability and transforming small probabilities into manageable large positive values

suitable for optimization algorithms.
Now, from equations 4.9 and 4.11 we obtain:

$$\varphi'(x_i, y_i, k_i, \bar{\theta}) = -\log \pi(y_i, k_i) + \frac{1}{2} \log |\Sigma(y_i, k_i)| + \frac{1}{2}(x_i - \mu(y_i, k_i))^T \Sigma(y_i, k_i)^{-1}(x_i - \mu(y_i, k_i)) \tag{4.12}$$

In our proposed method, we initialize the Gaussian Mixture Model using the output obtained from our proposed superpixel-based CNN method. To seamlessly integrate this information into our energy minimization framework, we employ a K-means clustering approach, dividing the data into two clusters corresponding to our classes of interest. This initial clustering forms the basis for our Expectation Maximization (EM) algorithm [Wu et al., 2008], consisting of two essential steps, the expectation step and the maximization step. In the expectation step, we leverage the prior model parameters, derived from the CNN output, along with the K-means clustered data, to estimate the likelihood of pixel classifications. By employing the GMM formula we discussed earlier, we iteratively refine these estimates in the maximization step. Here, the algorithm adjusts the GMM parameters, including the means, covariances, and mixing coefficients, to maximize the likelihood of the observed and estimated pixel classifications. This iterative process ensures the accurate segmentation of road and non-road pixels, making our methodology more robust and reliable for road segmentation tasks.
Having laid the foundation with our unary term derived from the Gaussian Mixture Model (GMM) using the meticulous Expectation Maximization (EM) algorithm and CNN initialization, our attention now turns to the crucial pairwise interactions. As depicted in Equation 4.6, the pairwise term for each pixel, denoted by ψ', captures subtle relationships between neighbouring pixels, enhancing the segmentation by promoting local coherence. Differing from the approach in [Rother et al., 2004], we introduced our smoothness energy based on the Potts model. In this formulation, label discrepancies between neighbouring pixels are penalized, shaping the segmentation result with spatial consistency. Mathematically our smoothness term is formulated as follows:

$$\psi'(y_i, y_j) = \begin{cases} \omega, & \text{if } y_i \neq y_j \\ 0, & \text{if } y_i = y_j \end{cases} \tag{4.13}$$

In this formulation, $\psi'(y_i, y_j)$ takes the value of ω, when neighbouring pixels y_i and y_j have different labels ($y_i \neq y_j$) and 0 when they share the same label ($y_i = y_j$). This calculation is performed for all neighbours of each pixel i. ω specifies the penalty for adjacent pixels having differing labels, reflecting the inherent cost of label discontinu-

ity. This penalty is subsequently scaled by β in the overall energy function, adjusting the relative influence of local consistency on the segmentation outcome.

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i, j-1)	(i, j)	(i, j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)

Figure 4.5: The two dimensional indices of the 8 neighbours pixels surrounding $p = (i, j)$.

In Traditional Iterated Conditional Modes (ICM) methods commonly employ 4 nearest neighbours for label smoothing. However, we recognized the limitations of this approach, particularly in achieving accurate optimization due to local convergence issues. To enhance the optimization accuracy, we extended our neighbourhood system to include 8 neighbouring pixels around each pixel. This expansion enables a more comprehensive assessment of the surrounding context, ultimately leading to improved segmentation performance. For each class, we count the number of of neighbour pixels, that their label do not match the label of the current pixel. These customized pairwise terms, outlined in Figure 4.5, add a valuable layer of context-awareness to our energy minimization process. To maintain the balance between smoothness and accuracy, the total smooth energy is penalized with a carefully chosen threshold value (β) , ensuring a nuanced approach to road segmentation. Now that the energy model is fully defined, the new segmentation $\hat{Y}$ can be estimated as a global minimum:

$$\hat{Y} = \arg\min_{Y} E'(Y, \bar{k}, \bar{\theta}, X) \tag{4.14}$$

Figure 4.6 showcases select image samples from the KITTI dataset [Fritsch et al., 2013a], where our enhanced ICM method has been applied to achieve seamless road segmentation. Notably, our adapted ICM method yields a significantly smoother segmentation along road border compared to our superpixel-based CNN approach, exemplifying the enhanced capabilities of our proposed methodology. The detailed values and implementation specifics will later be discussed in section 4.6.

4.5.2 Loopy Belief Propagation (LBP)

Performing probabilistic inference for large-scale multivariate random variables, such as in image segmentation, presents significant computational challenges. To facilitate an efficient method for the joint estimation of per-pixel object class labels, Loopy Belief Propagation (LBP), as introduced by Murphy et al. [Murphy et al., 1999], offers

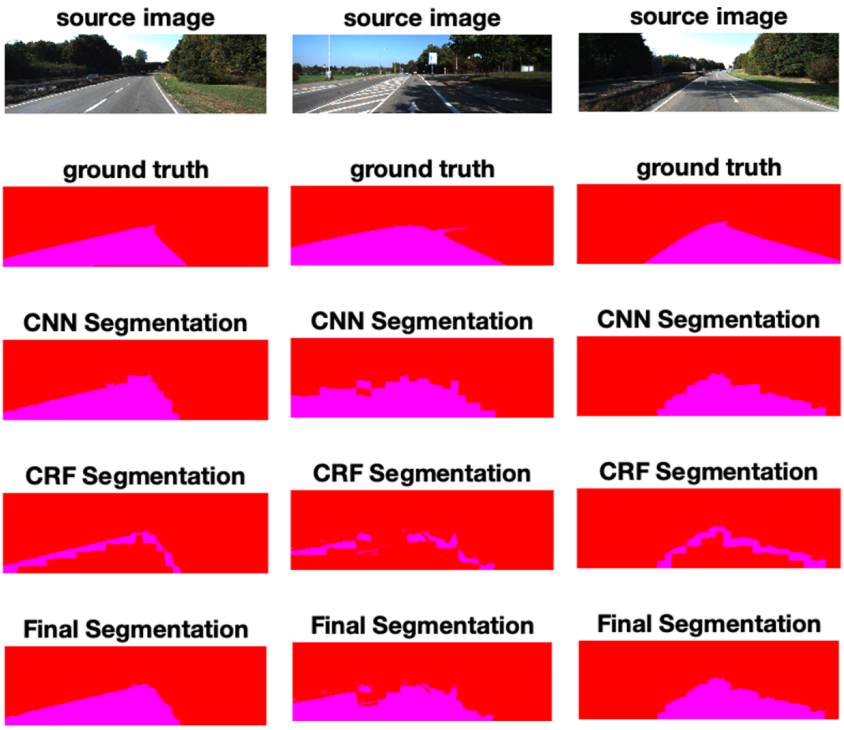


Figure 4.6: CRF-ICM based smoothed road segmentation result from KITTI validation set. The first horizontal row shows input images. Ground-truth images are shown in the second line. Third row shows our proposed SP-CNN segmentation results explained in chapter 3. Smoothed road segmentation results along road boundary obtained from our proposed ICM method are shown in fourth line. The last row presents the final labelling prediction.

an approximation of true marginals through iterative local computations of beliefs.

Loopy Belief Propagation (LBP) serves as a powerful technique for finding approximate solutions in Conditional Random Fields (CRFs), particularly effective in complex graph structures with loops. The process involves iterative updates and exchanges of messages among the nodes of the graph. In each iteration, every node, say node i, computes a "message", $m_{i \to j}(y_j)$, to be sent to an adjacent node, j. This message encapsulates node i's current estimate or "belief" about the probable label configuration, denoted as $b_i(y_i)$, which is formed from the incoming messages of all other neighbours except j.

In general, the LBP algorithm begins with an initialization phase where all messages are set to a constant, often 0 or 1, allowing for a uniform starting point. Messages are then passed iteratively, ensuring that each node sends a message to its neighbour only after all incoming messages have been received, except for the one from the destination node itself. The exact order of message passing is flexible and can be

tailored to the problem's specifics. Upon convergence, nodes formulate their beliefs by integrating all incoming messages.
Given the inherent loops within the graph, exact inference is unattainable, which is why LBP serves as an approximation technique. Guaranteeing convergence can be challenging, hence certain criteria are established such as stopping after a fixed number of iterations, or when the variation in energy becomes negligible.
The objective is to attain label assignments that maximize the joint probability through Maximum A Posteriori (MAP) estimation as articulated by [Nowozin et al., 2011]. Different message-passing algorithms like sum-product, max-product, and min-sum are utilized to compute the MAP estimation of variables in graphical models [Kschischang, 1999, Weiss and Freeman, 2001, Yedidia et al., 2003]. They each offer distinct methods of message calculation, hence influencing the nature of their approximations.

Sum-Product Message Passing: Which is also know as belief propagation, is used for calculating marginal probabilities. The algorithm adopts a comprehensive approach by taking the sum of the product of incoming messages and potentials, encapsulating a broader range of possibilities. Once all messages have been passed, each node can compute its belief. The message and belief update rules are as following:

$$\begin{aligned} m_{i\to j}^{(t)}(y_j) &= \sum_{y_i} \left(\exp(-\alpha\varphi(x_i, y_i)) \exp(-\beta\psi(y_i, y_j)) \prod_{k\in\mathcal{M}(i)\setminus j} m_{k\to i}^{(t-1)}(y_i) \right) \\ b_i^{(t)}(y_i) &= \exp(-\alpha\varphi(x_i, y_i)) \prod_{k\in\mathcal{M}(i)} m_{k\to i}^{(t)}(y_i) \end{aligned} \tag{4.15}$$

Here, $m_{i\to j}^{(t)}(y_j)$ represents the message from node i to node j concerning label y_j at iteration t. $\mathcal{M}(i)$ is the set of neighbouring nodes of i excluding j, and $m_{k\to i}(y_i)$ represents the message received by node i from its neighbour k. The unary and pairwise potentials within the energy function, $\varphi(\cdot)$ and $\psi(\cdot)$, respectively, contribute to these messages as described in Equation 4.3. The potential functions φ and ψ have already been characterized in terms of energy. When employing the Gibbs distribution within a probabilistic graphical model, we need to transform these energy values into probabilities through the Boltzmann distribution. This transformation entails exponentiating the negative energy term in the formula.

Max-Product Message Passing: Focusing on the most probable outcomes, the max-product algorithm simplifies the problem by maximizing the product of incoming messages and potentials. This strategy expedites computation but can be more

susceptible to local optima. The message and belief update rules in the max-product approach reformulated as follows:

$$\begin{aligned} m_{i\to j}^{(t)}(y_j) &= \max_{y_i} \left(\exp(-\alpha\varphi(x_i, y_i)) \exp(-\beta\psi(y_i, y_j)) \prod_{k\in\mathcal{M}(i)\setminus j} m_{k\to i}^{(t-1)}(y_i) \right) \\ b_i^{(t)}(y_i) &= \exp(-\alpha\varphi(x_i, y_i)) \prod_{k\in\mathcal{M}(i)} m_{k\to i}^{(t)}(y_i) \end{aligned} \tag{4.16}$$

Min-Sum Message Passing: The min-sum algorithm is a variation of the max-product algorithm, where operations are performed in the log domain. This converts products into sums and maximization into minimization, because we often work with negative log-probabilities (costs or energies). In this context, the "min" part is actually finding the minimum energy configuration, not minimum values.

$$\begin{aligned} m_{i\to j}^{(t)}(y_j) &= \min_{y_i} \left(\alpha\varphi(x_i, y_i) + \beta\psi(y_i, y_j) + \sum_{k\in\mathcal{M}(i)\setminus j} m_{k\to i}^{(t-1)}(y_i) \right) \\ b_i^{(t)}(y_i) &= \alpha\varphi(x_i, y_i) + \sum_{j\in\mathcal{M}(i)} m_{j\to i}^{(t)}(y_i) \end{aligned} \tag{4.17}$$

The difference between the three types of message passing methods—Sum-Product, Max-Product, and Min-Sum—lies in the way they compute messages and use them to obtain labels. The Sum-Product algorithm provides probabilistic information (marginal probabilities) about label assignments for each pixel, which represents the belief in each label's likelihood given the observed data. The Max-Product algorithm, which is closely related to the Min-Sum algorithm when potentials are in the log domain, aims to find the most probable single global label assignment (the MAP estimate) that minimizes the energy function. The Min-Sum algorithm, which is an efficient approximation of the Max-Product algorithm when working with log-domain potentials, also aims at identifying a single label assignment that minimizes the energy, albeit in a more computationally efficient manner due to the additive nature of its operations. Ultimately, the choice between sum-product and max-product (or min-sum) algorithms is dependent on the application's needs and the desired balance between computational efficiency and the robustness of the inference.
After the Loopy Belief Propagation algorithm has converged, or after a fixed number of iterations, a set of beliefs for each node will be obtained. The MAP estimates the best label assignment for each node i by choosing the label y_i^* that maximizes the belief at that node (for sum-product or max-product) or minimizes the energy (for

min-sum). For the sum-product and max-product algorithms, this can be formalized as:

$$\hat{y}_i = \arg\max_{y_i} b_i(y_i) \tag{4.18}$$

Here, $b_i(y_i)$ is the belief for node i, computed as previously described. For the min-sum algorithm, the label assignment would typically be represented as finding the label that minimizes the sum of costs:

$$\hat{y}_i = \arg\min_{y_i} b_i(y_i) \tag{4.19}$$

When assessing the performance of algorithms designed for probabilistic inference in graphical models, it is crucial to analyse their computational complexities. These complexities dictate the practicality of applying such algorithms to real-world problems like image segmentation.

The **Sum-Product** algorithm, which is utilized for computing marginal probabilities, stands out as the most computationally demanding in among of three. It requires summing over the product of messages for all label combinations, making its complexity higher. This algorithm's message passing complexity is $O(V \cdot D \cdot K^2 \cdot \text{degree})$, where V is the number of nodes, D is the dimensionality of the feature space, K is the number of labels, and *degree* is the average number of neighbours per node(eg. 4-Connectivity or 8-Connectivity). The time complexity, considering the iterative nature of the algorithm, expands to $O(V \cdot D \cdot K^2 \cdot \text{degree} \cdot \text{iterations})$.

On the other hand, the **Max-Product** and **Min-Sum** have lower computational complexity compared to Sum-Product, because they focus on finding the maximum or minimum value across the labels, rather than calculating a distribution. These algorithms sidestep the computational overhead of the feature space dimensionality D seen in Sum-Product, since they work directly on the label space and the messages are based on the pairwise potentials between labels and the current belief about the labels, rather than the features of the data itself. The unary potentials are considered in the calculation of the belief, but this is often considered a pre-computation step since it does not change between iterations. Their message passing complexity is generally noted as $O(K^2 \cdot \text{degree})$, with the time complexity following as $O(K^2 \cdot \text{degree} \cdot \text{iterations})$. These are typically lower than that of the Sum-Product algorithm per iteration, mainly due to not incorporating D. The node degree plays a pivotal role, influencing computational load through the number of messages exchanged. In tasks like road segmentation, higher degrees can improve accuracy but also demand more computational resources. Execution times for these algorithms can differ widely in practice, influenced by the graph's architecture and the efficiency of the algorithm's implementation.

Adaptive LBP $(X, Y, T, \text{MaxItr}, \alpha, \beta, \bar{\theta})$;

Input : Pixel features (X), initial labels from CNN (Y), convergence tolerance (T), maximum number of iterations (MaxItr), parameters of GMM $(\bar{\theta})$, penalize factors (α, β)
Output: Approximate MAP labeling

1) Initialize GMM parameters $\bar{\theta}$ with CNN outputs ; `// GMM Model Initialization`

2) Calculate unary potentials using GMM (See Algorithm 1); `// Unary Energy`

3) Initialize messages for all variables to zero; `// Message Initialization`

while *not converged and iteration count* $\leq$ *MaxItr* **do**

4) Update Messages using Min-Sum; `// Message Update`

$$m_{i\to j}(y_j) = \min_{y_i \in \text{all labels}} \Big[\alpha \cdot \varphi'(x_i, y_i, k_i, \bar{\theta}) + \beta \cdot \psi''(y_i, y_j) + \sum_{k \in \text{neighbors of } i \text{ except } j} m_{k\to i}(y_i)\Big]$$

5) Calculate Belief for each label; `// Belief Computation`

$$b_i(y_i) = \alpha\varphi'(x_i, y_i, k_i, \bar{\theta}) + \sum_{z \in \text{neighbors of } i} m_{z\to i}(y_i)$$

6) Check for convergence based on change in messages and T ;

7) Update MAP assignment ; `// MAP Assignment and Energy Calculation`

$$\hat{y}_i = \arg\min_{y_i} b_i(y_i)$$

end

Algorithm 2: Our proposed Enhanced LBP for Road Segmentation using Min-Sum Message Passing with GMM

In this research, we introduce a novel adaptation of Loopy Belief Propagation (LBP) combined GMMs specifically tailored for semantic road segmentation. Detailed in Algorithm (2), our method innovatively initializes and integrates GMM components within the LBP framework, focusing particularly on areas along road borders to enhance computational efficiency.
Central to our approach is a unique strategy for initializing GMM parameters, leveraging the discriminative capabilities of a previously developed SP-CNN method. This ensures, that the GMM parameters are semantically meaningful and resulting in significantly enhanced unary potentials within the CRF. This targeted initialization, applied specifically to superpixels adjacent to road borders, allows for more efficient processing compared to traditional methods that operate on entire images.
To optimize our model further, we chose the min-sum method for message passing within our CRF model, aiming to minimize the energy function as described in equation 4.6. Our in-depth computational complexity analysis which is discussed earlier, revealed that both the Min-Sum and Max-Product methods, with complexities of $O(K^2 \cdot \text{degree} \cdot \text{iterations})$, are more efficient than the Sum-Product algorithm. Although our experiments employed both Min-Sum and Max-Product methods with nearly identical results in accuracy and computational time, we present here the Min-Sum method.
The model initiates with messages set to a default value of zero. Similar to our adaptive ICM method, as discussed in section 4.5.1, we implement the Potts model with minor modifications. The pairwise potential reformulated as:

$$\psi''(y_i, y_j) = \begin{cases} \omega, & \text{if } y_i \neq y_j \\ \epsilon, & \text{if } y_i = y_j \end{cases} \tag{4.20}$$

Here, $\psi''(y_i, y_j)$ assumes the value of ω, when neighbouring pixels y_i and y_j have differing labels. In contrast to ICM, we deliberately incorporate the ϵ value, when $y_i = y_j$. This inclusion is intended to mitigate numerical issues, such as division by zero or logarithm of zero, which are common in algorithms like Min-Sum due to very small probability values that could lead to numerical instability. By establishing ϵ as a lower bound for these probabilities, we maintain numerical stability by ensuring that the values aids in the convergence of the algorithm by preventing extreme probability values.
In the context of ICM, the primary focus is on local updates and immediate energy reduction. This approach often necessitates considering the labels of neighbouring nodes in a greedy manner. However, ICM does not explicitly require considering cases where neighbouring nodes share the same label, as its primary concern is whether a change in a single node's label reduces the total energy.
In contrast, LBP's message-passing mechanism inherently accounts for both scenarios: when neighbouring nodes have identical or different labels. The introduction of the ϵ value in this setting has a direct impact on the probabilities involved, adjusting the penalties or rewards for label consistency or inconsistency, thereby influencing the overall labelling decisions.

The LBP is inherently iterative, continuing until minimal energy change is observed or a predefined number of iterations is reached. Our message-passing sequence was selected to be right, left, up, down, any sequence considering 4-connectivity neighbourhood to keep the computational efficiency close to our adaptive ICM method. This choice of connectivity effectively balances computational load, while maintaining segmentation accuracy. Details on hyper-parameter settings will be discussed in Section 4.6.
Empirical results from applying our hybrid method to the KITTI dataset [Fritsch et al., 2013a] have been promising. As Figure 4.7 demonstrates, our approach not only adeptly captures the interplay between global and local pixel interactions, but also enhances road segmentation accuracy, particularly along road borders.

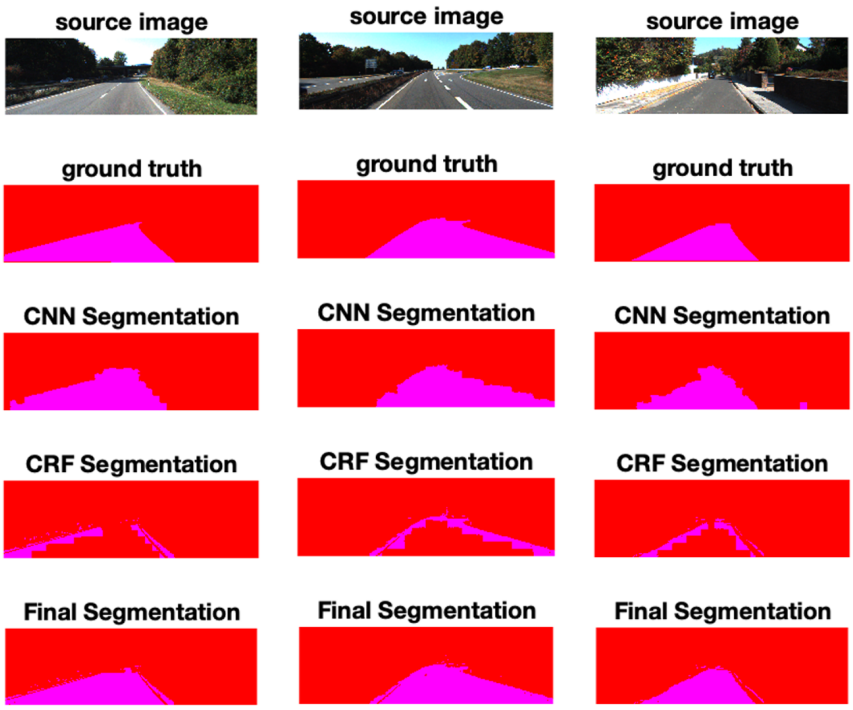


Figure 4.7: CRF-LBP based smoothed road segmentation result from KITTI validation set. The first horizontal row shows input images. Ground-truth images are shown in the second line. Third row shows our proposed SP-CNN segmentation results explained in chapter 3. Smoothed road segmentation results along road boundary obtained from our proposed method are shown in fourth line. The last row presents the final labelling prediction.

4.5.3 DenseCRF with Mean-Field Approximation

In our research, we employed another specialized inference method to achieve effective and smooth road segmentation. This method utilizes a fully connected Conditional

Random Field (CRF), as detailed in [Krähenbühl and Koltun, 2011]. In this model, every pixel in an image is considered to be connected to every other pixel, forming a *fully connected* network. The key to this model's efficiency lies in its use of "pairwise edge potentials", which describe the relationship or dependency between each pair of connected pixels. Uniquely, these potentials in the model are formulated as Gaussian linear combinations. To manage the complexity inherent in this fully connected network, the approach incorporates the concept of 'mean-field' approximation. Rather than computing the direct influence of every other pixel, which is computationally intensive, the mean-field approximation assumes that each pixel is influenced by a collective, average effect of the entire image. This theoretical construct significantly reduces the computational burden, making it feasible to analyse and predict the labels in images.

Contrary to the approach in the referenced paper where unary potentials are obtained from TextonBoost, our method utilizes class conditional probability maps generated from the softmax layer of our specifically designed SP-CNN model. These probability maps are integrated into the framework of the fully-connected CRF as described in [Krähenbühl and Koltun, 2011], with the aim of improving road segmentation outcomes. This novel application harnesses the sophisticated output of our SP-CNN model to guide the final CRF pixel-wise labelling, with a particular focus on the challenging road border areas. By integrating the probability class maps, we ensured a robust and nuanced starting point for the CRF segmentation process, effectively leveraging deep learning insights for initial pixel classification. In addition, our preprocessed initialization is particularly advantageous, as it brings the mean-field approximation closer to optimal solutions at the outset, reducing the iterative burden and enhancing the overall convergence speed.

Our method aligns with the established pairwise potentials and Gaussian kernel formulation as in [Krähenbühl and Koltun, 2011] (equation 4.21):

$$\psi_{i,j} = \omega_1 \exp(-\frac{\|\boldsymbol{P}_i - \boldsymbol{P}_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}) + \omega_2 \exp(-\frac{\|\boldsymbol{P}_i - \boldsymbol{P}_j\|^2}{2\sigma_\gamma^2}) \tag{4.21}$$

Here, two Gaussian kernels are employed to define the pairwise potentials, the appearance kernel and the smoothness kernel. The appearance kernel is designed to assess the colour similarity between adjacent pixels, factoring in both their spatial positions ($\boldsymbol{P}_i$ and $\boldsymbol{P}_j$) and colour intensities (I_i and I_j). This kernel's primary role is to encourage pixels with similar colours and proximities to be classified with the same label. In contrast, the smoothness kernel focuses exclusively on the spatial positioning of pixels. Its purpose is to promote label consistency among spatially close pixels, effectively reducing the presence of small, isolated regions in the segmentation output. The extent of these kernels' influence is finely tuned by the parameters σ_α and σ_β, which control the degrees of colour and spatial nearness, and σ_γ, which sets the threshold for defining the size of small isolated areas. The equation is balanced by linear combination weights ω.

Furthermore, the algorithm leverages the mean-field approximation and a message passing framework within a fully-connected graph to efficiently approximate the latent variables that minimize the Gibbs energy for labelling. As demonstrated in Figure 4.8, the application of this sophisticated CRF method on selected images from the KITTI dataset results in notably smoother road segmentation boundaries.

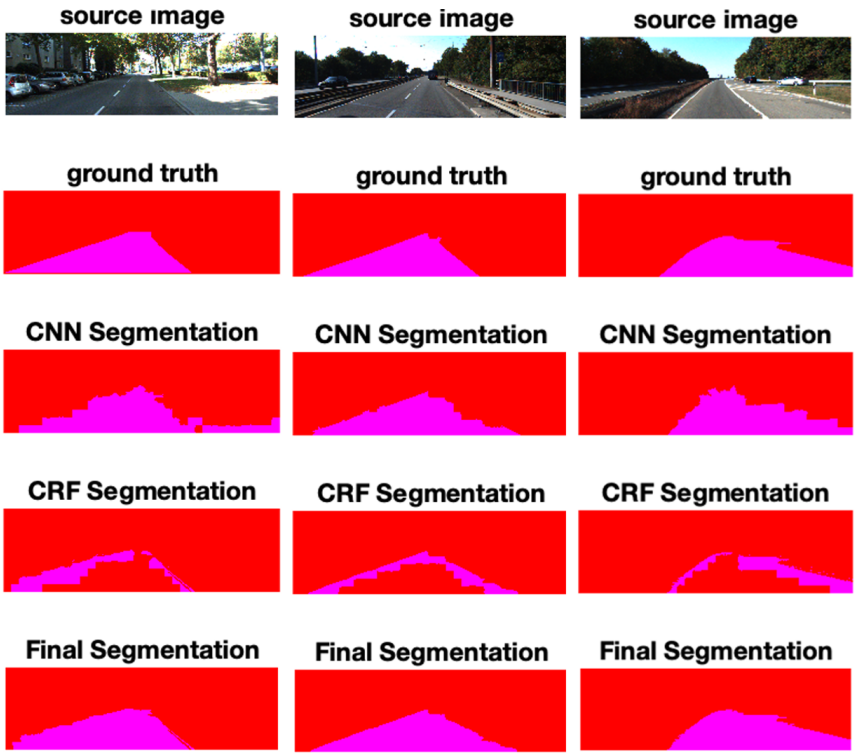


Figure 4.8: CRF-Meanfield based smoothed road segmentation result from KITTI validation set. The first horizontal row shows input images. Ground-truth images are shown in the second line. Third row shows our proposed SP-CNN segmentation results explained in chapter 3. Smoothed road segmentation results along road boundary obtained from our proposed method are shown in fourth line. The last row presents the final labelling prediction.

This approach simplifies the complexity of traditional loopy belief propagation (LBP) message passing from quadratic to linear, enhancing the processing of long-range dependencies and the ability to delineate fine edge details.The mean-field algorithm allows to approximate the maximum posterior efficiently from a billion edges of fully connected CRF [Krähenbühl and Koltun, 2011].
A pivotal aspect of this method lies in its ability to simplify the computational complexity of message passing, contrasting starkly with traditional loopy belief propagation (LBP). In standard LBP, the message-passing complexity is quadratic ($O(n^2)$) due to the requirement for each node in a fully connected graph to exchange mes-

sages with every other node. This quadratic complexity arises from the fact that for a graph with n nodes, each node must communicate and update its state based on the states of $n-1$ other nodes, leading to an extensive number of of $n \times (n-1)$ pairwise interactions.
Conversely, the DenseCRF model utilizes the mean-field approximation, where the update of each node's belief is influenced by an efficiently computed global aggregation of effects from all other nodes in the graph. This is achieved through the application of Gaussian filtering, which aggregates these influences in a way that scales linearly with the number of nodes. As a result, instead of each node processing interactions individually with every other node, the mean-field approximation allows the update for each node to be computed in linear time ($O(n)$), relative to the number of nodes. Such an optimization significantly enhances the algorithm's capability in handling long-range dependencies and in delineating fine edge details, crucial for high-fidelity image segmentation tasks [Krähenbühl and Koltun, 2011].

4.6 Evaluation and Discussion

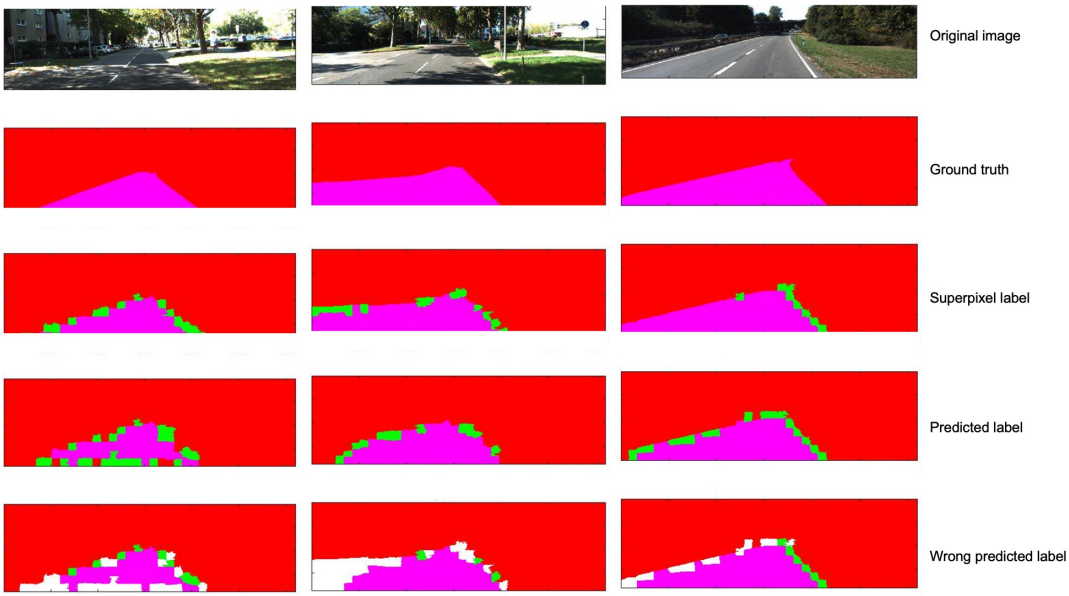
The entirety of our experiments was executed on an *Intel(R) Core(TM) i7-4790K CPU at 4GHz*, providing a robust computational foundation to support our analyses. In this chapter, we further evaluate our method using the publicly available KITTI dataset (referenced in Chapter 3.4.1), renowned for its urban scenario images. The dataset encompasses 502 8-bit RGB images, categorized into training, validation, and test sets, each annotated for three distinct semantic classes: urban markings (UM), multiple urban markings (UMM), and unmarked urban streets (UU). Specifically, the training set includes 289 images, while the test set comprises 290 images. These images vary in dimensions, primarily within the width range of $[1226, 1238, 1241, 1242]$ and height range of $[370, 374, 375, 376]$. For validation purposes, 20% of the training set images, ensuring diversity across categories and sequences, were selected. To ensure continuity with the methodologies outlined in our prior research (see Chapter 3), we diligently maintained the same experimental conditions and parameter specifications. This included the deployment of SLIC parameters set to $N_{sp} = 400$ and $m_{slic} = 35$, resulting in 396 superpixels per image. These superpixels were then organized into an 11×36 lattice, which was subsequently utilized as the input for our CNN model.
As elucidated in 3.5, our evaluation within the superpixel framework is predicated on establishing ground truth labels according to the majority pixel label within each superpixel. In chapter 3, Superpixels where the dominant label does not constitute at least an 80% majority are designated with an *unlabelled* status, thereby creating a three-class system, *road*, *non-road*, and *unlabelled*. However, for pixel-level analysis on the validation set, superpixels marked as *unlabelled* were combined with the *non-road* class to form a singular negative class, facilitating a binary classification evaluation scheme in line with the KITTI road benchmark's two-class evaluation system. In contrast to the ternary system employed in the training and validation phases of the previous chapter, this chapter adopts a binary classification approach from the start, for all datasets. This simplification enhances our method's ability to distinguish

between 'road' and 'non-road' categories more effectively. The impact of this adjustment is evident in the segmentation discrepancies highlighted in figure 4.9. Areas that were previously categorized as 'unlabelled' superpixels have a significant influence on our predictive pixel-based segmentation outcomes, often leading to false positives or negatives in pixel classification. The shift from a ternary to a binary classifier has notably mitigated these inaccuracies, thereby refining our prediction accuracy.

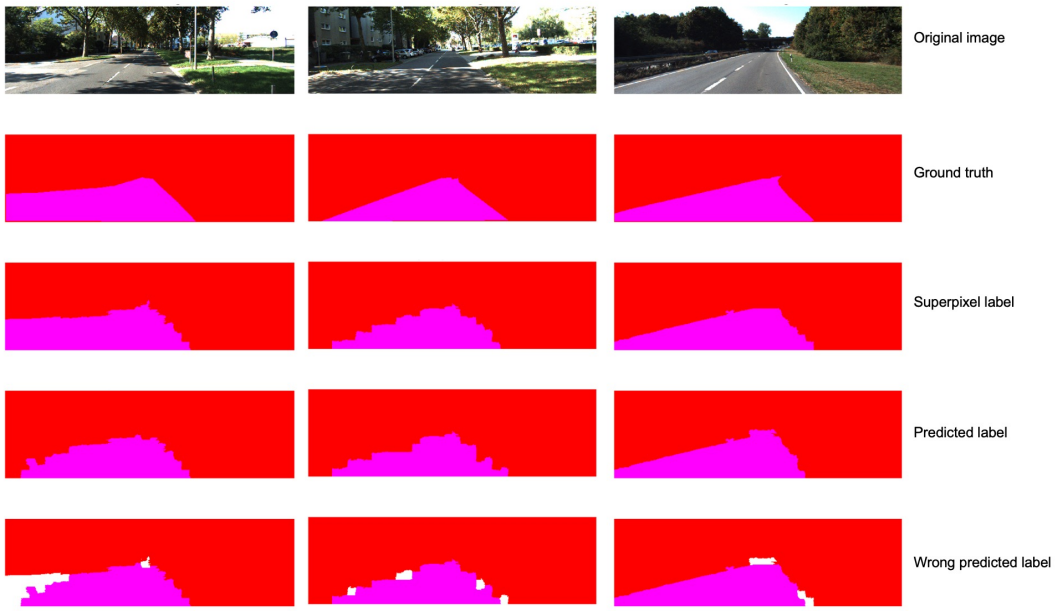
4.6.1 Evaluation on Image Perspective

In current work, our primary objective is to enhance the road segmentation results obtained from our previous work. The evaluation was conducted both in the image perspective and in the bird's-eye-view projection provided by the KITTI dataset. Due to the unavailability of the ground truth for the KITTI dataset's test data (as noted in [Fritsch et al., 2013a]), we utilized the validation set to enable a comprehensive evaluation of our methods from an image perspective. Initially, we rigorously assessed the results yielded by each of the three proposed CRF optimization techniques, ICM, LBP, and dense CRF with mean-field method, on this validation set. This process involved a comparative analysis to determine which technique demonstrated the highest accuracy in alignment with our initial method described in the previous chapter. Following this internal evaluation, the technique that exhibited superior performance was then selected for further validation. This advanced step involved applying our chosen CRF optimization method on the test set and subsequently submitting these results to the KITTI benchmark website. This approach ensured our adherence to the evaluation protocols established by KITTI (referenced in [Fritsch et al., 2013a]), allowing for a direct and fair comparison with other state-of-the-art methods.

In our study, Fully connected CRF based on mean-field approximation has the best performance. The weight of the appearance kernel in Dense CRF method was set at $\omega_1 = 0.1$, and the kernel widths were determined as $\sigma_\alpha = 60$ and $\sigma_\beta = 10$. Additionally, we assigned the parameters $\omega_2 = 1$ and $\sigma_\gamma = 3$. In the Iterated Conditional Mode (ICM) approach, which exhibited a marginally lower accuracy, we rigorously tested various values for beta, ranging from -1 to 20, maintaining $\omega = 1$. Optimal performance, as validated against ground truth on the training set, was achieved with potential weights of $\alpha = 1$ and $\beta = 20$ (refer to Equation 4.6). In contrast, Loopy Belief Propagation underperformed compared to the other two methods. The best results for this approach were obtained using potential weights of $\alpha = \beta = 1$, $\omega = 0.95$, and $\epsilon = 0.05$. It is noteworthy that for all three CRF methods, we consistently used 20 iterations. The results derived from each of these CRF techniques, with the aforementioned parameters, are detailed in a dedicated section. One exemplary outcome, illustrating the comparative effectiveness of all three CRF methods, is presented in Figure 4.10.



(a) Ternary SP-CNN Classifier



(b) Binary SP-CNN Classifier

Figure 4.9: contrast between ternary and binary classification in Superpixel-Based CNN approach as discussed in Chapter 3. Red indicates 'non-road' pixels, pink signifies 'road' pixels, green represents 'unlabelled' pixels, and white denotes incorrectly predicted pixels.

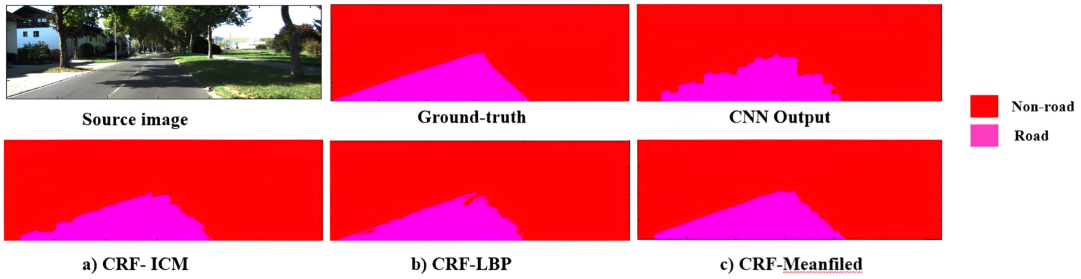


Figure 4.10: Smoothed road segmentation based on a) Iterated Conditional Mode,b) Loopy Belief Propagation, c)Fully connected CRf with mean-field approximation method.

Method	ACC	MaxF	PRE	REC	FPR	FNR
CNN Ternary-Classifier	96.50%	90.08%	92.80%	88.07%	7.20%	2.65%
CNN Binary-Classifier	94.41%	94.28%	91.41%	97.34%	8.59%	2.60%
CRF_LBP	94.31 %	87.58 %	88.58 %	86.74 %	9.51 %	4.92%
CRF_ICM	96.90 %	87.94 %	88.02 %	93.62 %	16.95 %	0.89 %
CRF_MeanField	**96.85** %	91.47 %	92.30 %	90.65 %	7.70 %	2.10 %

Table 4.1: This table presents the evaluation results on the KITTI validation set, comparing the performance before and after the application of various CRF techniques, including ICM, LBP, and Dense CRF based on the mean-field approximation. Additionally, the results for the CNN Ternary-Classifier, as discussed in Chapter 3, are based on a three-class label system, road, un-road, and unlabelled. In contrast, the CNN Ternary-Classifier here focuses solely on two classes, road and un-road.

Table 4.1 summarizes the average evaluation results on the validation set for all urban categories based on the three adaptive CRFs. The accuracy obtained from CNN part was 94.41%. The variance in evaluation values on validation sets, reported across the two chapters (3 and 4), highlights the influence of different classification strategies on model evaluation. In previous chapter, we employed a three-class system (road, non-road, and unlabelled) for the CNN model, resulting in an accuracy of 96.50% and a maximum F1 (MaxF) score of 90.08%. This approach, which classified uncertain pixels as *unlabelled*, offered a detailed assessment, but at the expense of a slightly lower F-measure score due to its conservative nature, while boosting overall accuracy. In contrast, in this chapter, a binary classification (road and non-road) was

adopted, simplifying the evaluation. This change led to an accuracy of 94.41% and a higher F-measure score of 94.28% , driven by a notable increase in recall (97.34%). This suggests, that the binary system was more effective in identifying road pixels, rather than ternary classifier. The observed results from applying CRF techniques, specifically the CRF Mean-Field method, show a higher overall accuracy 96.85%, but a lower F-measure score 91.47% compared to the CNN model's performance. The CNN Binary-Classifier has a lower precision (91.41%) but a higher recall (97.34%) compared to the CRF_MeanField method (92.30% precision and 90.65% recall). This suggests, that the CNN Binary-Classifier is better at identifying most of the relevant road pixels (high recall) but at the cost of incorrectly labelling more non-road pixels as road (lower precision). Conversely, CRF_MeanField is slightly more precise in labelling road pixels but misses more actual road pixels (lower recall). The observed outcome, particularly the lower precision of the CRF methods compared to the CNN model, can potentially be explained by this smoothing process's influence on the superpixels at the boundaries. The reclassification during the CRF smoothing step, intended to refine the edges, might inadvertently alter the labels of some superpixels incorrectly. This behaviour is shown in figure 4.11.

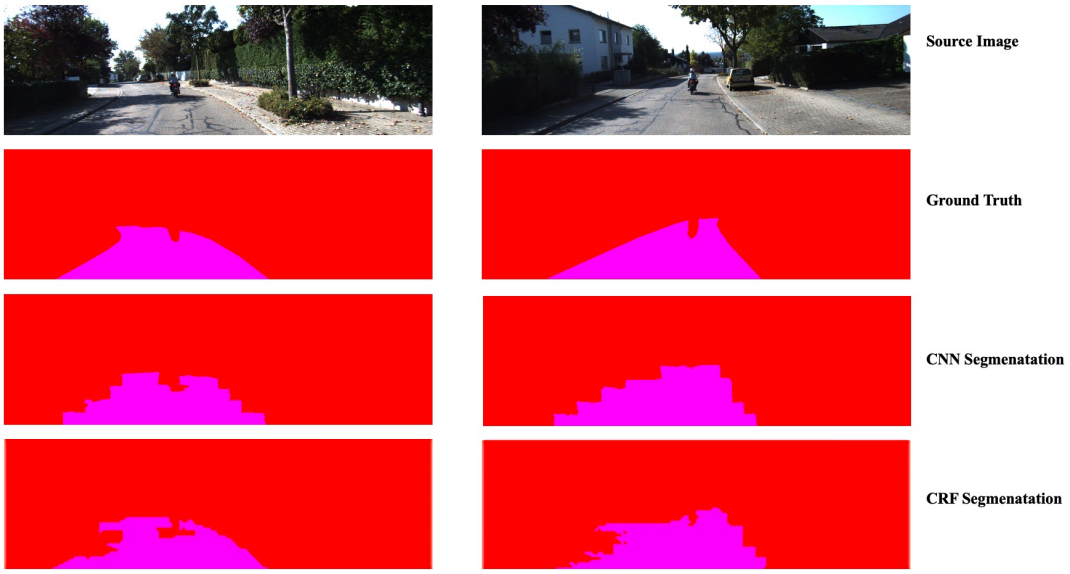


Figure 4.11: Balancing Precision and Recall in Road Segmentation. The third-row images showcase the CNN segmentation, achieving a high recall that captures most of the road, despite some imprecision along the boundaries. The bottom-row images detail the CRF segmentation, which offers refined boundary precision and smoother contours but at the trade-off of reduced recall, evidenced by its omission of certain road segments that the CNN identified.

This phenomenon suggests, that while the CRF methods are effective in creating visually smoother boundaries, they might also introduce errors by re-misclassifying non-road elements as road, especially in complex boundary regions, where the distinction between road and non-road is less clear. The result is a higher overall accuracy due to the improved consistency along borders, but at the expense of precision. Thus, the trade-off here is between achieving smoother, more visually consistent road boundaries and maintaining the precision of road classification. The CRF methods excel in the former, but face challenges in the latter.

4.6.2 Evaluation on Birds Eye Perspective

In our road segmentation study, the evaluation of the test set was conducted using the KITTI benchmark's birds-eye view (BEV) perspective. This method involves projecting the images onto the ground plane based on the known camera geometry, a standard procedure in the KITTI benchmark evaluation. The results of this evaluation, as showcased in Table 4.2, are systematically categorized into four distinct road types: UM, UMM, UU, and URBAN by KITTI. These classifications, integral to the KITTI benchmark's assessment framework, allow for a detailed analysis of the model's performance across various urban road scenarios.
The evaluation results, along with illustrative figures, are provided at the end by the KITTI benchmark website for the test set. This external validation on a diverse set of test images is crucial for an unbiased assessment of the model's capabilities. In Figure 4.12, we present three samples from the test set evaluation in both image and BEV perspectives, demonstrating the effectiveness of our model in segmenting streets, while also highlighting areas of false detections, typically in regions where shadows cover the street.

In the bird's-eye view (BEV) perspective of our evaluated road segmentation images, the areas marked in red (false negatives) and blue (false positives) appear notably larger compared to the standard image perspective. This is a result of the perspective warping process required to transform these images into BEV. During this transformation, objects and regions closer to the camera get stretched more significantly to represent a top-down view. Consequently, small misclassified areas in the original image, such as shadows or road edges, become more prominent and enlarged in BEV. This stretching effect is essential to provide a comprehensive overhead view but can exaggerate the size of error regions, highlighting the model's segmentation challenges, especially along road boundaries.
In this comparative analysis, the segmentation along road borders is markedly smoother, with the elimination of the stair-step artifacts, that were previously observed. This refinement is particularly evident in high-contrast areas where road surfaces meet grass or curb edges. Our current methodology demonstrates a heightened precision in delineating the road from adjacent non-road elements. This is especially prominent in the third image of the set, where regions surrounding the ego-lane are densely pop-

Benchmark	**M**axF	**A**P	**P**RE	**R**EC	**F**PR	**F**NR
UM_ROAD	83.22 %	72.94 %	77.11 %	90.39 %	12.23 %	9.61 %
UMM_ROAD	90.96 %	84.63 %	87.86 %	94.29 %	14.32 %	5.71 %
UU_ROAD	80.02 %	67.93 %	77.56 %	82.64 %	7.79 %	17.36 %
URBAN_ROAD	85.97 %	77.81 %	82.04 %	90.31 %	10.89 %	9.69 %

Table 4.2: This table presents the evaluation results on the KITTI test set for the Mean-Field CRF method. The evaluation metrics used here, MaxF (Maximum F-Measure) and AP (Average Precision), are those prescribed by the KITTI benchmark. MaxF represents the highest F1 score (the harmonic mean of precision and recall) achieved across different decision thresholds, indicating the optimal balance between precision and recall. AP measures the average precision achieved across varying recall levels.

ulated with stones. Despite the complex texture, our algorithm accurately segments these areas, a testament to the sophistication of the approach.
The first image highlights the challenges posed by shadows on the road surface, which can lead to false positives or negatives in detection. Nevertheless, the transition from road to grass and curb remains distinctly captured, showcasing an improvement in detection under varied lighting conditions.
In the second image, we observe highly accurate road segmentation, albeit with minor instances of false negatives. Notably, the proportion of false positives is somewhat elevated compared to the other two images, yet the delineation remains refined, devoid of the stair-shaped artifacts characteristic of our previous models. This particular artifact correction is noteworthy. The areas with a similar texture to the road, had been erroneously classified as part of the road by the SP-CNN method due to textural challenges—demonstrate our method's evolution. Our initial approach struggled to exclude such textures from the ego road due to their close resemblance to the road surface. The introduction of the Conditional Random Field (CRF) method has been pivotal in this context. By focusing on pairwise potentials and enforcing consistency among neighbouring pixels, the CRF method has inadvertently led to a more generous interpretation of what constitutes the road. This approach has resulted in a smoother, more contiguous road segmentation, albeit at the cost of increasing false positives. These false positives manifest as an artifact of the CRF's inclination to homogenize the textural features, thereby integrating these regions into the road segment. This outcome underscores the nature of the CRF method in our segmentation process. While it has enhanced the smoothness of the segmentation contours, it has

also reintroduced previously excluded areas back into the road category due to their textural resemblance to the road surface.

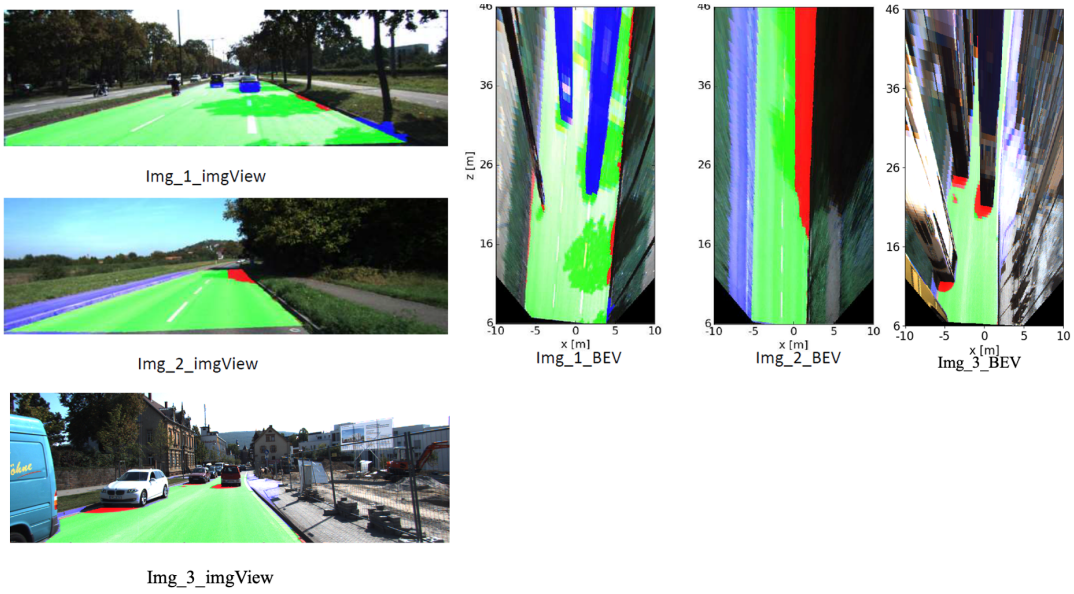


Figure 4.12: Road segmentation result from official KITTI test set in baseline and bird eye view perspectives. Here, red denotes false negatives, blue is false positives and green represents true positives.

In the comprehensive evaluation of road segmentation models using the KITTI benchmark test set, a comparison between our SP-CNN method (See Table.3.3) and the CNN with CRF-Mean Field smoothing (See Table.4.2) reveals notable improvements in segmentation performance. Overall, across the four categories of UM_ROAD, UMM_ROAD, UU_ROAD, and URBAN_ROAD, the integration of CRF smoothing has led to an average increase of 3.17% in the MaxF score and 4.08% in the Average Precision (AP). These enhancements highlight the effectiveness of CRF in refining the segmentation quality, particularly in terms of achieving a balanced precision and recall (as indicated by the MaxF score) and consistently improving precision across varying recall levels (as reflected in the AP score).
When we focus on the UMM_ROAD category specifically, the benefits of CRF smoothing become even more apparent. In this category, there is an improvement of approximately 5% compared to our CNN classifier and the MaxF score rises from 85.07% to 90.96%, and the AP from 79.86% to 84.63%. This significant improvement in both metrics suggests, that the CRF method is particularly adept at delineating

road areas in more complex urban and multi-lane settings. The higher MaxF score indicates a strong balance between accurately identifying road pixels (precision) and capturing most of the actual road areas (recall). These results underscore the efficacy of CRF smoothing in enhancing the model's ability to distinguish between road and non-road areas, especially in challenging urban environments. The ability of the CRF-enhanced model to maintain high precision without significantly sacrificing recall demonstrates its potential in applications, where accurate road segmentation is critical.
In our study, we carefully analysed the variance in accuracy levels between the birds-eye view (BEV) projection (Table 4.2) and the image perspective evaluation (Table 4.1). This analysis revealed, that the BEV projection initially showed marginally lower accuracy compared to the image perspective. A significant factor contributing to this discrepancy is the inherent diversity and complexity of the test set, which features challenging scenarios such as prevalent shadows and detailed curbstone patterns on road surfaces, posing obstacles to accurate segmentation. While initially BEV projection demonstrated weaker accuracy, primarily due to the way errors are amplified in this perspective, our focused refinement efforts significantly improved its performance. Errors originating from inaccurately defined superpixels along the road borders, which in BEV tend to expand over a larger area, were particularly addressed. By enhancing the precision of superpixel segmentation, especially at road boundaries, we effectively mitigated the spread of these errors.

4.6.3 Run-time Analysis

This section presents a comprehensive comparative analysis of our semantic segmentation approach against several state-of-the-art methods, highlighting runtime efficiency. This comparative study is crucial to demonstrate the practicality and performance of our approach, particularly in environments with significant hardware constraints.
Our method effectively employs superpixels in conjunction with a streamlined CNN network to generate coarse pixel-based segmentation. This is refined by an optimized pixel-wise CRF technique, applied selectively to area around the road contour, achieving an ideal balance between accuracy and computational efficiency. Our approach, optimized for CPU-based systems, contrasts with other methods dependent on high-powered GPUs.
In Table 4.3, we delve into a detailed comparison of our semantic segmentation approach with some of the top-performing methods in the field. Notably, *LODNN* [Caltagirone et al., 2017], is a high-end method operating on the *NVIDIA GTX980Ti 6 GPU*, showcasing exceptional accuracy and efficiency in segmentation. It stands out with an impressive MaxF score of 96.05%, which is about 5% higher than our final approach and achieves a rapid runtime of just 0.018 seconds, highlighting the advantages of advanced GPU processing. However, this high performance comes with the need for substantial computational resources, as indicated by its AI-Score of 16038. This score indicates that *LODNN*'s processing capabilities are approximately 11.46 times greater than that of our CPU-based system.

While other methods like *UP_CONV_POLY* [Oliveira et al., 2016] and *DDN* [Mohan, 2014] also demonstrate high MaxF scores (surpassing ours by nearly 3%), they similarly rely on powerful GPU hardware, such as the *NVIDIA Titan X*, to achieve their fast processing times. These methods, although efficient and accurate, underscore the dependence on high-end GPUs for optimal performance.
In contrast, our CPU-based approach, even without CRF integration, achieves a MaxF of 85.07% and an AP of 79.86% with a remarkable runtime of only 0.019 seconds on an *Intel Core i7-4790K CPU* with the lower AI-Score of 1400, indicating significant efficiency for a less powerful system. Integrating CRF further enhances our method's MaxF to 90.96% and AP to 84.63%, albeit with a slightly longer runtime of 0.21 seconds. Overall, our method demonstrates a balanced trade-off between accuracy and runtime efficiency, making it a compelling choice for systems with limited hardware capabilities.

Method	**P**rocessor	**M**axF	**A**P	**R**untime(s)	**A**I-Scores
LODNN	NVIDIA GTX980Ti GPU, 6GB memory	96.05 %	95.03%	0.018	16038
UP_CONV_POLY	NVIDIA Titan X GPU.	95.52 %	92.86	0.083	24870
DDN	NVIDIA GTX980Ti GPU, 6GB memory	94.17 %	92.70 %	2	16038
Ours (without CRF)	Intel(R) Core(TM) i7-4790K CPU @4GHz	85.07 %	79.86 %	**0.019**	1400
Ours (With CRF)	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	**90.96** %	84.63 %	**0.21**	**1400**

Table 4.3: Comparative Performance of SP-CNN and CRF-Refined methods on the KITTI UMM test set, against advanced approaches. only results of published methods are reported. LODNN: [Caltagirone et al., 2017], UP_CONV_POLY [Oliveira et al., 2016], DDN [Mohan, 2014].AI-Scores:https://ai-benchmark.com/ranking_deeplearning.html

4.7 Conclusion

This chapter has detailed an cohesive approach for fine-grained semantic segmentation, enhancing our previous work, that synergized superpixels with convolutional neural networks (CNN). We refined this approach to improve segmentation maps derived from a superpixel-based CNN for urban scene segmentation tasks, making it particularly suitable for applications in robotics and related embedded systems that

demand real-time visual scene understanding.
A key focus of our work was to enhance segmentation accuracy, especially at critical area such as road boundaries, while maintaining low execution times and computational demands, utilizing cost-effective resources. Our methodology merged the robust capabilities of CNN with Conditional Random Fields (CRF), a probabilistic graphical model extensively applied in structured prediction tasks within Computer Vision, including semantic segmentation. Initially, our superpixel-based CNN model provided dense, albeit coarse, scene labelling. We then integrated the potential derived from the CNN into a sophisticated CRF model, aimed at refining these segmentation outcomes. Recognizing the necessity for real-time system applicability, we embarked on a thorough examination of three adaptive CRF techniques. This exploration was driven by the need to find a harmonious balance between accuracy and operational efficiency. Our analysis culminated in the selection of a particular CRF solution that demonstrated superior performance, efficiently addressing the challenges of semantic segmentation.
The algorithm we developed, strikes a well-balanced trade-off between accuracy and efficiency.It is not only suited for real-time CPU-based applications, but also shows commendable results when compared with top-performing GPU-based CNN models, particularly in the KITTI dataset, a well-known public benchmark for semantic road segmentation. Notably, our approach achieves these promising outcomes with significantly lower memory and computational requirements, underscoring its viability and effectiveness in resource-constrained environments.

5 Enhancement of the superpixel-CNN based road segmentation using semi-supervised Modified-CycleGAN

In our quest for advanced semantic segmentation solutions, we previously explored methods heavily reliant on comprehensive supervision and the incorporation of pairwise dependency modelling. While these approaches have their merits, they are hampered by the high costs associated with acquiring extensive annotated data and their limited adaptability to new, unseen data scenarios. Despite the utility of Conditional Random Fields (CRFs) in improving road segmentation with manageable computational demands, we identified persistent challenges in accurately capturing complex elements such as unpaved roads, shadows, grass, and side-walks. These challenges arise from either the scarcity of training data or the inherent limitations of CRF models, particularly in terms of pairwise potential.
Recognizing these challenges, we propose a refined strategy using a modified version of the Cycle Generative Adversarial Network (CycleGAN) [Zhu et al., 2017]. Our approach significantly improves adaptability, enabling the capture of the underlying distribution of data beyond the fixed constraints of traditional CRF models. This adaptability results in enhanced generalizability, making our method more robust against variations in data, including unseen scenarios. Our strategy leverages the principles of unpaired image-to-image translation to refine initial semantic segmentation predictions, with a specific focus on street scenes crucial for automated mapping. Our contributions include a streamlined generative model based on CycleGAN with reduced parameters, an innovative "RGB-L" dataset (standard RGB channels with an additional 'L' channel for segmentation labels), and an enhanced CycleGAN objective function. Preliminary results on the KITTI benchmark demonstrate an improvement in segmentation performance. Experimental results on the KITTI public road segmentation benchmark demonstrate a 4-7% improvement in accuracy over our previous superpixel-CNN approach, achieving comparable performance with top-performing algorithms in recent un/semi-supervised semantic segmentation tasks.

This chapter presents portions of the research, that have previously been disseminated in the studies referenced as [Zohourian and Pauli, 2022b] and [Zohourian and Pauli, 2022a].

5.1 Introduction

Rapid and precise determination of pixel labels in semantic segmentation is crucial for real-time systems like road detection in autonomous driving, where accurate interpretation of road environments is vital for ensuring vehicle safety and operational efficiency. Various methods for semantic segmentation have predominantly depended on a) comprehensive supervision, involving extensive training images with detailed pixel-level annotations, and b) the incorporation of pairwise dependency modelling to refine segmentation outcomes. These methods, thoroughly explored in preceding chapters, as highlighted in seminal works [LeCun et al., 2015, Schmidhuber, 2015, Krizhevsky et al., 2017, Farabet et al., 2013, Long et al., 2015, Krähenbühl and Koltun, 2011]. While effective to a certain extent, these approaches face significant challenges in terms of data acquisition costs and adaptability to new, unseen scenarios.

In our previous work, we have utilized Conditional Random Fields (CRFs) to enhance road segmentation capabilities with manageable computational demands. However, these traditional CRF methods encountered difficulties in accurately capturing complex elements like unpaved roads, cast shadows, grass, and side-walks adjacent to road boundaries. These challenges primarily arose from insufficient training data or inherent limitations within the CRF models, especially in terms of their pairwise potential capabilities. The original CRF models struggled with capturing long-range dependencies among pixels in different regions of the image [Lafferty et al., 2001, Krähenbühl and Koltun, 2011, Murphy et al., 1999, Rother et al., 2004]. Attempts to incorporate hierarchical connectivity and higher-order potentials into region-based CRF approaches led to slight improvements, but the models remained computationally intractable and lacked sufficient accuracy [Kohli et al., 2009, Ladickỳ et al., 2009]. Fully connected CRF models addressed some of these issues, but at a significantly higher computational cost [Krähenbühl and Koltun, 2011].

To overcome these limitations, we propose a novel approach utilizing a modified version of the Cycle Generative Adversarial Network (CycleGAN) [Zhu et al., 2017], which enforces object connectivity and spatial relationship dependency consistency without being limited to a very specific class of pairwise potential. This method demonstrates a remarkable capacity to adaptively learn and capture the underlying distribution of data, surpassing the fixed constraints of traditional CRF models. The adaptability of this approach significantly enhances its generalizability, making it robust against variations in data, including scenarios not encountered during training. This approach is critical for maintaining high efficiency in real-time system integration, especially for street scenes crucial in automated mapping.

Our approach introduces a streamlined CycleGAN, optimized with a reduced parameter set to enhance performance and reduce computational demands. Additionally, we have innovated by introducing a unique "RGB-L" dataset. Each image in this dataset comprises standard RGB channels augmented with an additional 'L' channel carrying segmentation labels derived from our SP-CNN segmentation technique. This enriched dataset provides our network with comprehensive information, facilitating a more informed and accurate learning process. Furthermore, we have refined the CycleGAN's objective function by integrating a supplementary term that calcu-

lates the Manhattan distance (L_1) between a subset of our RGB-L images and their corresponding targeted labels. This modification serves as a guiding mechanism for the network's learning process, ensuring the generation of semantically accurate images with higher fidelity. By limiting the adversarial learning procedure to the road boundaries predicted in our recent work and utilizing a limited number of annotated images, we have significantly boosted segmentation performance. To summarize, our current approach makes the following pivotal contributions:

1. **Modified-CycleGAN Model:** Introduction of a Modified Cycle Consistency Generative Adversarial Network, which significantly improves road segmentation results derived from our superpixel-based CNN in a semi-supervised setting.

2. **"RGB-L" Dataset Innovation**: Utilization of the novel "RGB-L" dataset for advanced contextual and spatial analysis. The proposed adversarial method enforce cycle consistency to learn the mapping between an unpaired "RGB-L" dataset and a label domain. The full architecture is shown in Fig. 5.1.

3. **Optimization for Computational Efficiency:** This optimization involves two main alterations. First, the redesigning of the CycleGAN's residual blocks to reduce the complexity, and second limitation of the adversarial learning procedure to previously predicted road boundaries to boost segmentation performance.

4. **Objective Function Refinement:**Optimization of the CycleGAN's objective function with an added L_1 loss component between a subset of paired images and their corresponding targets, ensuring more precise and accurate segmentation.

The remainder of this chapter is organised as follows. In Section 5.2, we explore the latest advancements in semantic segmentation within the deep learning domain, highlighting the role of adversarial learning-based approaches in this field. This section serves to contextualize our work within the broader spectrum of current research and technological developments. Following this, Section 5.3 offers an in-depth overview of Generative Adversarial Networks (GANs) as a pivotal element in unsupervised machine learning. This section aims to lay the foundational and theoretical groundwork essential for understanding the subsequent contributions detailed in this chapter. It includes an exploration of the domain translation technique, its key components, and an examination of various GAN architectures. In Section 5.4, we provide a detailed description of our proposed method. This section covers the entire framework, beginning with a brief explanation of our superpixel-based CNN approach, which forms the core concept for our semantic segmentation task. We then delve into the specifics of our modified-CycleGAN, designed to enhance the results of our segmentation efforts. Section 5.5 is dedicated to elucidating the implementation and training aspects of our proposed adversarial model. This section is crucial for understanding the practical application and operational nuances of our model. Our analysis of the experimental results, focusing on both accuracy and time-efficiency, is presented in Section 5.6.

This section evaluates the effectiveness of our proposed method, supported by empirical data and comparative analysis. The chapter concludes with Section 5.7, where we summarize our findings.

5.2 Related Works

The effectiveness of segmentation models is often compromised by various factors such as occlusions, changes in illumination, extensive paved areas, shadows, and the overlapping of objects, all of which hinder the generalization capabilities of these models. Despite the considerable success of networks based on fully convolutional architectures [Long et al., 2015, Ronneberger et al., 2015, Badrinarayanan et al., 2017] and the use of Conditional Random Fields (CRFs) for enhanced semantic segmentation [Chen et al., 2014, Farabet et al., 2013, Papandreou et al., 2015, Schwing and Urtasun, 2015, Zheng et al., 2015], their performance is highly contingent on maintaining a consistent data distribution between training and testing datasets. However, the real-world scenario often presents a disparity between these datasets. Moreover, most cutting-edge methods are fully supervised and thus require a substantial amount of labeled training data to achieve optimal performance. Although data augmentation is a common strategy to increase the volume of training data, the disparity in data distribution between training and testing sets can likewise lead to suboptimal results. To mitigate this issue and enhance the adaptability of methods for better generalization, unsupervised techniques have emerged as a powerful tool. These techniques aim to improve the generalizability of deep learning models to new image domains without relying on labeled data in the target domain [Isola et al., 2017, Zhu et al., 2017] and without integrating specific relational dependencies or higher-order terms directly into the model, unlike in CRF-based approaches [Zheng et al., 2015, Krähenbühl and Koltun, 2011]. The objective is to learn the hidden distribution of data without the need for labeled data. Generative Adversarial Networks (GANs)[Goodfellow et al., 2020] represent a class of unsupervised learning where the goal is to generate new samples from an unlabeled distribution. GANs consist of two networks, namely the Generator (G) and the Discriminator (D), which are trained concurrently in an adversarial manner to implicitly discover the underlying distribution of the training examples. Various types of GANs have been proposed, including Conditional GANs (cGANs) [Mirza and Osindero, 2014], Deep Convolutional GANs (DCGANs) [Radford et al., 2015], and Pix2pix [Isola et al., 2017]. DCGANs utilize deep convolutional and convolutional-transpose layers in the discriminator and generator, respectively, to learn from unlabeled image data. cGANs [Mirza and Osindero, 2014] use images as conditional inputs, feeding them into both the generator and discriminator. Pix2pix [Isola et al., 2017], an extension of the cGAN architecture, employs a U-Net-based [Ronneberger et al., 2015] network as the generator and the PatchGAN [Isola et al., 2017] architecture as the discriminator. Pix2pix has evolved with the introduction of CycleGAN [Zhu et al., 2017], which uses a cycle consistency loss for domain translation without requiring paired data.
Recently, various GAN-based post-processing approaches have been developed to achieve smoother road semantic segmentation results, primarily for aerial images.

Luc [Luc et al., 2016] proposed a convolutional semantic segmentation network coupled with an adversarial network to surpass the performance of traditional networks. StreetGAN [Hartmann et al., 2017] trains on arbitrarily-sized road patches in aerial images through a GAN network to analyze and enhance attributes in challenging road extraction areas. A modified cycle GAN proposed in [Cho et al., 2020] aims to improve semantic segmentation for low-light images. A Dual-Hop Generative Adversarial Network (DH-GAN) [Costea et al., 2017] first segments roads and intersections in aerial images, followed by a smoothing-based graph optimization procedure to fit an optimal road graph. In [Zhang et al., 2019], a Multi-conditional Generative Adversarial Network (McGAN) is introduced to refine road topology and generate complete road network graphs. McGAN is designed to handle multiple conditions or factors simultaneously, encompassing various aspects of road topology such as structure, connectivity, and surrounding features, leading to more accurately refined road topology in the generated images.

5.3 Fundamentals of Generative Adversarial Nets(GANs)

In this section, we delve into the fundamentals of Generative Adversarial Networks (GANs), starting with an overview of their core principles and architecture. Our exploration begins with the conceptual framework and architecture of GANs, where we provide a detailed explanation of the components and mechanisms that define GANs, including their discriminators, generators, and pivotal loss functions. We then transition to examining the technical challenges of GANs, addressing common hurdles such as *Mode Collapse* and *Non-Convergence* encountered during GAN training. The final subsection focuses on key variants of GANs, where we discuss three significant GAN-based models such as cGAN, Pix2pix, and CycleGAN. This comprehensive overview aims to provide a clear understanding of GANs' theoretical and practical aspects, thereby establishing a solid base for a more profound understanding of the proposed approach detailed in the following sections.

5.3.1 Conceptual Framework and Architecture of GANs

Generative Adversarial Networks (GANs) [Goodfellow et al., 2020], a unique subset of deep neural networks, fall under the broader category of generative models. Diverging from discriminative models, which are tailored to distinguish between different classes, GANs and other generative models are designed to learn the distribution of input data through unsupervised learning. This characteristic allows GANs to create new, synthetic data samples that closely mimic the original input data.
In GANs, as depicted in figure 5.1, the adversarial process involves two main components, a generator (G) and a discriminator (D). The generator $G(z;\theta_g)$, a multilayer perceptron with learnable parameters θ_g, generates synthetic data $G(z)$ from a random input noise vector z drawn from a distribution $p(z)$. The 'latent' space, represented by Z, serves as the basis for generating new data samples. The discriminator network $D(y;\theta_d)$, another multilayer perceptron with learnable parameters θ_d, evaluates whether the given samples are real. It processes both real data samples y

from the real data distribution $p_{data}(y)$ and synthetic data $G(z)$ from the generator, outputting a probability estimate $D(y)$ for real data and $D(G(z))$ for generated data. For simplicity and ease of discussion, subsequent references to the generator and discriminator functions will omit the explicit mention of their parametrization by θ_g and θ_d, respectively. Thus, $G(z)$ will refer to the output of the generator, when applied to noise vector z, and $D(y)$ will denote the discriminator's output for a given data sample y, without explicitly detailing the underlying parameters.
The adversarial relationship between G and D drives the learning in GANs. The generator's goal is to closely mimic real data in the domain Y to 'deceive' the discriminator, while the discriminator aims to distinguish real data in Y from synthetic data generated from Z. This dynamic resembles a contest between counterfeiters (generator) and detectives (discriminator).
The objective function of GANs, defining this adversarial game, is formalized in a minimax equation, where the optimal generator G^* and discriminator D^* are derived from the following optimization:

$$\begin{aligned} G^*, D^* &= \arg\min_G \max_D V(G, D) \\ V(G, D) &= \mathbb{E}_{y \sim p_{data}(y)}[\log D(y)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))] \end{aligned} \tag{5.1}$$

In this equation, $\mathbb{E}$ represents the expectation, and $V(G, D)$ is the value function, that G seeks to minimize and D aims to maximize. To train G, D is held constant, focusing only on synthetic data from G. This setting implies that G is trained to minimize $\log(1 - D(G(z)))$, effectively tricking D into classifying the synthetic data as real. According to Goodfellow et al. [Goodfellow et al., 2020], the GAN reaches an equilibrium when the generator (G) creates data so realistic that the discriminator (D) cannot tell if it is real or synthetic, ultimately assigning a probability of 0.5 to all samples. This leads to a 50% accuracy rate, which indicates total uncertainty.

5.3.2 Technical Challenges of GANs

The two most common challenges faced by GANs during training are Mode Collapse and Non-Convergence.
Mode collapse is a problem that can occur in GANs during training, where the generator produces limited variations of the same output, rather than producing a diverse set of outputs. Generative models should be able to capture all modes of the target data distribution. When the GAN fails to do this, mode collapse occurs. This occurs when the generator learns to exploit a weakness in the discriminator and produces samples that the discriminator is unable to distinguish from real samples. As a result, the generator stops producing new and diverse samples, leading to a collapse of the diversity of the generated samples. If the mode starts to collapse, the similarity of generated images increases . It means, probably there is an image or a small sub-set of images, that minimize loss for the generator and the generator maps each input to that point only. mode collapse is common when the generator and discriminator

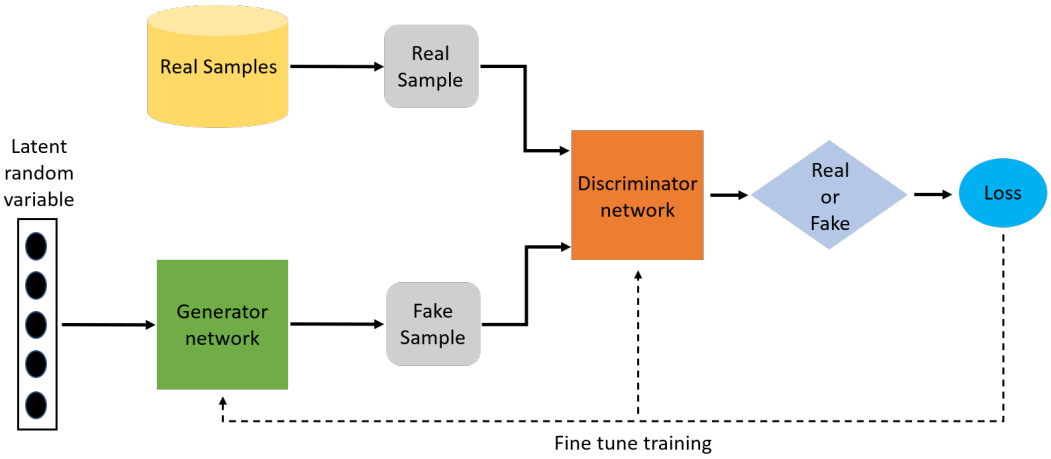


Figure 5.1: Basic Architecture of a Generative Adversarial Network (GAN), featuring two primary components, a Generator and a Discriminator. The Generator creates samples from a source of random noise, often referred to as the 'latent' space, while the Discriminator evaluates these samples alongside real data, learning to differentiate between real and fake. This iterative and competitive training process underlines the adversarial nature of GANs, essential for generating increasingly realistic images. Figure from [Dash et al., 2021].

architectures are not well-designed or when the training dataset is not diverse enough. Non-convergence is another problem that can occur in GANs during training, where the generator and discriminator do not reach an equilibrium, and the training process never reaches a steady state. This can occur for a variety of reasons, such as poor choice of network architecture, poor hyper-parameter selection, or poor quality of the training data. Non-convergence can also happen if the generator and discriminator are not well-matched in terms of capacity or if the training process is not properly regularized. This leads to unsatisfactory results, in which the generated images are not able to imitate real images and by spending more time on training, the model does not improve.

In addressing the prevalent issues of mode collapse and non-convergence in GAN training, several strategies can be employed for more effective training. Firstly, employing a diverse dataset is crucial to ensure a wide range of training samples, which

helps in avoiding over-fitting to specific patterns. Secondly, the selection of an appropriate architectural design for the neural network is key, as it must align with the learning objectives and complexities of the model. Thirdly, integrating robust regularization techniques is important to stabilize the training process. Lastly, using an effective loss function can significantly guide and improve the model's learning trajectory towards successful convergence. These combined strategies are instrumental in enhancing the overall efficacy of GAN training.
This thesis directly confronts these challenges in two key subsections. The Data Preparation subsection (5.5.1) outlines our approach to data input augmentation, enhancing the diversity and complexity of training data, which is crucial for preventing over-fitting and mode collapse. Additionally, the Model Analysis subsection (5.6.1) delves into the performance and stability of the model, providing insights that are key to ensuring effective convergence. These sections collectively underscore the strategic measures taken to mitigate common challenges in GAN training.

5.3.3 Types of Generative Adversarial Networks

Since its inception in 2014 by Ian Goodfellow et al. [Goodfellow et al., 2020], the original GAN architecture has evolved considerably. What began as a relatively basic framework has been extensively modified and refined, adapting to a myriad of tasks and applications. This progression underscores GANs' remarkable flexibility and their growing significance in deep learning. Numerous GAN variants have emerged, each tailored to particular data types, tasks, or to surmount specific training hurdles. This section delves into some of these variants, especially those most pertinent to our proposed methodology.

cGAN

In traditional GANs, reaching the global optimum often proves difficult, as the discriminator typically excels at distinguishing between real and synthetic images, outperforming the generator's performance. Additionally, the primary goal in these GANs is usually to trick the discriminator, rather than to ensure the generated data meets specific domain requirements or characteristics. This focus might not suit certain applications, like autonomous driving, where the generator's role is to create highly accurate, domain-specific imagery for simulating varied driving conditions, including diverse road environments or changes in weather and lighting. Here, the emphasis shifts from deception to the precision and relevance of the generated content, marking a significant deviation from the conventional GAN framework.
Conditional Generative Adversarial Network (cGAN), an extension of GANs, addresses some inherent limitations in the basic GAN framework. cGANs condition both the generator $G(z|x;\theta_g)$ and discriminator $D(y|x;\theta_d)$ on additional information x, such as class labels or other modalities, which correlates with the training examples. This method directs the generated data $G(z|x)$ more precisely towards the intended domain. The discriminator $D(y|x)$, in turn, evaluates the authenticity of the data y, taking into account the same conditional information x, which allows for a more

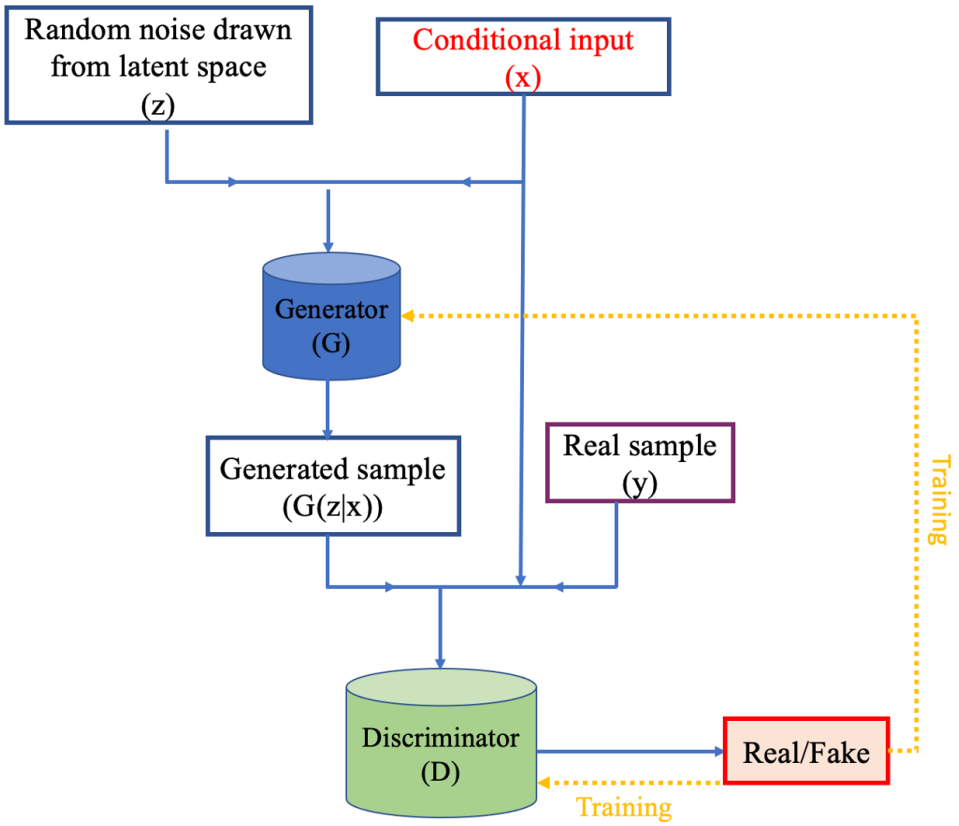


Figure 5.2: Illustration of the training procedure of a cGAN model. Adapted from [Mirza and Osindero, 2014] and [Mathwork.com].

nuanced assessment of the generated data's relevance to the desired domain. Figure 5.2 illustrates the architecture of a basic conditional adversarial network, showcasing how it integrates additional conditional elements into its structure. The objective function of cGAN is defined as:

$$\begin{aligned} G^*, D^* &= \arg\min_G \max_D V(G, D) \\ V(G, D) &= \mathbb{E}_{y \sim p_{data}(y)}[\log D(y|x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z|x)))] \end{aligned} \tag{5.2}$$

In cGAN, the generator G and discriminator D not only learn to create and evaluate data based on the latent space z and the real data distribution $p_{data}(y)$, but also take into account the conditional data x. This integration results in producing more targeted and domain-specific outputs, shifting the focus from merely deceiving the discriminator to generating data that adheres to specific criteria or domains.

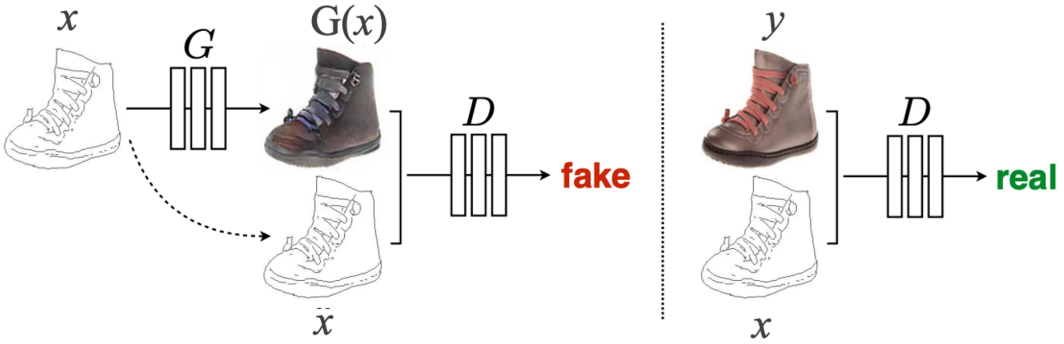


Figure 5.3: Architecture of the Pix2Pix model (Source: [Isola et al., 2017]). This model demonstrates a conditional GAN, that is specifically trained to convert edge into photorealistic images. The discriminator, D, is trained to distinguish between pairs of fake images generated by the generator and real edge-photo pairs. Simultaneously, the generator, G, aims to produce images that are realistic enough to mislead the discriminator. Notably, both G and D have access to the input edge maps during the training process.

Pix2Pix

GANs conditioned on an input image can be used for image-to-image translation. Image-to-image translation problems often map an image to a different image. These networks can be trained supervised [Isola et al., 2017], semi-supervised [Gan et al., 2017], or unsupervised [Zhu et al., 2017].
Pix2Pix, a specific form of a conditional Generative Adversarial Network (cGAN), is designed for a supervised image-to-image translation task. In Pix2Pix, the generator G transforms a conditional input image x into an output image $G(x)$, which aims to closely resemble the target image y in the training dataset. Figure 5.3 shows the Pix2Pix model architecture. Instead of mapping a random noise vector to an image, the Generator in Pix2Pix maps an image to another representation of the same image. The discriminator D in Pix2Pix evaluates the authenticity of the generated images. It examines pairs of images, where one pair consists of the real combination x and y, and the other pair comprises the conditional input x and the generated image $G(x)$. The discriminator's role is to distinguish between these real and generated pairs.
The objective function of Pix2Pix combines the cGAN loss with an L_1 loss. The cGAN loss ensures that the generated images are realistic, while the L_1 loss maintains a pixel-level similarity between the generated $G(x)$ and target y images. Drawing from the previously defined cGAN framework in equation 5.2, the objective function for the Pix2Pix model is articulated as follows:

$$L_{\text{Pix2Pix}}(G, D) = L_{cGAN}(G, D) + \lambda \cdot \mathcal{L}_{L_1}(G) \tag{5.3}$$

Where L_{cGAN} and $\mathcal{L}_{L_1}$ are calculating as following:

$$\begin{aligned} L_{cGAN}(G,D) =& \mathbb{E}_{x,y\sim p_{data}(x,y)}[\log D(x,y)]+ \\ & \mathbb{E}_{x\sim p_{data}(x),z\sim p(z)}[\log(1-D(x,G(z|x)))] \\ \mathcal{L}_{L_1}(G) =& \mathbb{E}_{x,y\sim p_{data}(x,y)}[||y-G(x)||_1] \end{aligned} \tag{5.4}$$

In this equation, $\|\cdot\|_1$ represents the L_1 norm, which calculates the absolute pixel-wise difference between the generated image $G(x)$ and the target image y. The expectation $\mathbb{E}_{x,y}$ is over the distribution of the data. The cumulative objective for the Pix2Pix model is thus expressed as:

$$G^*, D^* = \arg\min_G \max_D L_{cGAN}(G,D) + \lambda \mathcal{L}_{L_1}(G) \tag{5.5}$$

Here, the generator G is tasked with minimizing the combined objective function, striving to generate images that closely match the target images in a pixel-wise sense. Conversely, the discriminator D aims to maximize the same function, focusing on accurately distinguishing between real and generated image pairs. The parameter λ in the equation plays a crucial role as a balancing factor, harmonizing the adversarial nature of the GAN loss with the pixel-level accuracy enforced by the L_1 loss. This balanced approach enables the Pix2Pix model to effectively generate high-quality, realistic images that are closely aligned with their corresponding targets.
The generator in the Pix2Pix model utilizes the U-Net architecture, as detailed in [Ronneberger et al., 2015]. This architecture fundamentally comprises an Encoder-Decoder framework, enhanced with skip connections that link equivalent layers in the Encoder to their counterparts in the Decoder. A distinctive feature of the Pix2Pix model during the testing phase is that the generator operates similarly to how it does during the training phase. This means that techniques like Dropout are still utilized, and batch normalization relies on statistics from the test batch instead of those from the training batch. The discriminator, confronts both the source(conditional) input image (x) and the target image (y). It evaluates, whether the target image (y) is a plausible transformation of the source image (x). This setup enables the discriminator to understand the relationship between the input and output images, thereby obligating the generator to learn and replicate this relationship as well. The discriminator examines two pairs of images, one pair consisting of the input image (x) and the target image (y), and the other comprising the input image (x) and the generated image ($G(x)$).
The Pix2Pix model introduces an innovative discriminator architecture known as PatchGAN, specifically designed to enhance the detailing of high-frequency components in generated images. High-frequency details pertain to the finer aspects, such as edges and textures, which are crucial for the sharpness and realism of the image.

On the other hand, low-frequency components, which the L_1 loss effectively captures, relate to the broader and smoother areas, like large uniform regions and background colours. While the L_1 loss ensures that the overall structure and colors align well with the target image, it often falls short in preserving the finer details, leading to a somewhat blurred effect in the generated images.
PatchGAN addresses this challenge by evaluating smaller, patches of size $P \times P$ from the input image. Unlike a Fully Convolutional Network (FCN) that assesses the entire image in one go, PatchGAN's patch-based approach concentrates on the high-frequency elements within these patches. Each patch is analysed independently to determine its authenticity, focusing on the sharpness and texture details. This process is applied in a convolutional manner across the entire image, with the final discriminator output being an aggregation of the decisions made for each patch. This methodology ensures that the generator is not only accountable for the general resemblance but also for the replication of fine details throughout the image. Consequently, PatchGAN fosters the generation of images with more precise and realistic textures and edges, overcoming the common issue of detail loss associated with low-frequency focused approaches. The patch-based strategy also results in a more efficient discriminator, characterized by fewer parameters and faster operation compared to those assessing the entire image.

CycleGAN

In considering the various GAN techniques discussed earlier, it's important to acknowledge the challenges associated with preparing paired datasets, a necessity for supervised methods like Pix2pix. This process is resource-intensive and often impractical for real-world applications, highlighting the need for alternative image-to-image translation techniques that operate effectively without paired data. To address this, CycleGAN [Zhu et al., 2017], offers a novel approach for Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. CycleGAN seeks to match the performance of Pix2pix but without requiring paired data, utilizing an adaptive cycle constraint in its optimization process for translating between two domains.

In Pix2Pix, a supervised domain translation method, corresponding image pairs (x, y) from domains X and Y are necessary. The generator $G : X \to Y$ in Pix2Pix translates an image x from domain X to a corresponding image y in domain Y. In this setup, the number of samples in both domains X and Y is equal, as each image x from domain X is paired with a corresponding image y in domain Y. Conversely, CycleGAN achieves similar translations but operates under a different paradigm, handling unpaired data. In CycleGAN, the datasets from domains X and Y do not necessarily have an equal number of samples, denoted as N and M respectively. This reflects the unpaired nature of the data in CycleGAN, where the model learns to translate between domains X and Y without relying on one-to-one correspondence between individual samples. CycleGAN introduces an additional generator $F : Y \to X$ for reverse direction translation, facilitating a *back-translation* process to evaluate the fidelity of the translation. As illustrated in Figure 5.4, CycleGAN features a symmet-

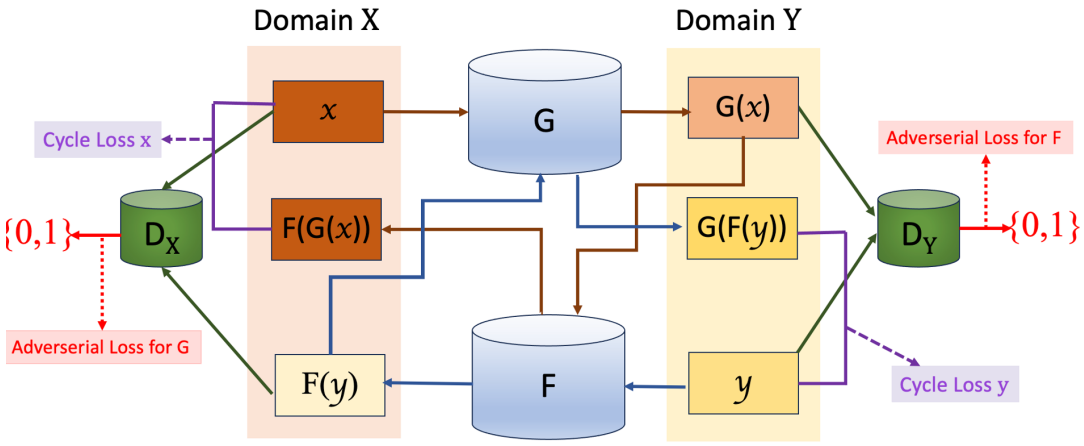


Figure 5.4: CycleGAN model architecture. The model includes two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, along with their associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate images from domain X into outputs that are indistinguishable from domain Y, and vice versa for D_X and F. Two cycle consistency losses are introduced for further regulation of these mappings. Cycle-consistency loss Y ensures that $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and cycle-consistency loss X ensures that $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. Adapted from [Zhu et al., 2017] and [Ji et al., 2020].

ric structure with two generators, G and F. Generator G maps inputs from domain X to Y ($G_{X \rightarrow Y}$), while generator F performs the reverse ($F_{Y \rightarrow X}$). Accompanying these generators are two discriminators, D_Y and D_X. Discriminator D_Y differentiates between images from domain Y and those translated by G from X (i.e., $G(X)$), and D_X performs a similar role for domain X and translations by F from Y (i.e., $F(Y)$). The ideal outcomes in a well-functioning CycleGAN model are $D_Y(y) \approx 1$ and $D_Y(G(X)) \approx 0$. These responses would indicate that D_Y is effectively distinguishing between real images from domain Y and generated images from $G(X)$, while G is continuously striving to generate images that are increasingly difficult for D_Y to classify as fake. The model aims to reach a point where G improves to such an extent that $D_Y(G(X))$ gradually approaches 1, signifying that the generated images are nearly indistinguishable from real images.
In the standard architecture of CycleGAN, each generator comprises an encoder that downsamples the input, followed by a series of layers for feature transformation and processing. These layers, often referred to as "transformer block" within this context, typically include residual blocks that aid in maintaining essential content during the transformation. After processing the features, a decoder upsamples them to construct the output image. The combination of these components allows the CycleGAN generators to learn complex mappings from the input to the output domain. On the discriminator side, the networks are fully convolutional, classifying sections of the

image, known as patches, as real or fake. This patch-based approach enables a more nuanced assessment of image authenticity, differentiating the CycleGAN from other architectures that evaluate the image in its entirety.
In order to overcome the unpaired image-to-image translation problems and to preserve key attributes between the input and the translated image, CycleGAN incorporates two key loss components, *Adversarial Loss* for aligning the distribution of generated images with the target domain, and *Cycle Consistency Losses* for ensuring fidelity in the translations. The adversarial losses for each mapping function are:

$$\begin{aligned} L_{adv}(G, D_Y, X, Y) =& \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \\ & \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \end{aligned} \tag{5.6a}$$

$$\begin{aligned} L_{adv}(F, D_X, Y, X) =& \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \\ & \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))] \end{aligned} \tag{5.6b}$$

Adversarial losses alone are insufficient to ensure that the mappings G and F in CycleGAN accurately translate an input to the desired output. To address this, CycleGAN employs cycle consistency loss, incorporating both forward and backward consistency terms. This loss ensures that an image can be translated from one domain to another and then back again, closely resembling its original form. It prevents contradictions in the mappings G and F and maintains the integrity of the image translation process across domains. The cycle consistency loss is defined as follows:

$$\begin{aligned} L_{cyl}(G, F) = & \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \\ & \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \end{aligned} \tag{5.7}$$

The overall CycleGAN objective function is:

$$\begin{aligned} L_{\text{CycleGAN}}(G, F, D_X, D_Y) =& L_{adv}(G, D_Y, X, Y) + \\ & L_{adv}(F, D_X, Y, X) + \\ & \lambda L_{cyl}(G, F) \end{aligned} \tag{5.8}$$

Here, λ balances the adversarial loss and the cycle-consistency loss. The training optimizes this function:

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} L_{\text{CycleGAN}}(G, F, D_X, D_Y) \tag{5.9}$$

In the upcoming sections, we will delve into the final contribution of this thesis, which is inspired by the principles of image-to-image translation and the CycleGAN framework. We propose a novel method that serves as a post-processing technique. This method is specifically designed to enhance the initial semantic segmentation predictions outlined in Chapter 3, with a focus on refining road segmentation results. Additionally, it addresses and resolves certain limitations identified in our earlier work. This approach exemplifies the practical application of advanced image translation concepts in improving the accuracy and reliability of segmentation tasks.

5.4 Proposed Method: Superpixel-CNN Enhancement with Semi-Supervised Modified-CycleGAN

This section outlines our comprehensive method, which integrates the Superpixel-based Convolutional Neural Network (SP-CNN) discussed in Chapter 3, with a Semi-Supervised Modified-CycleGAN technique to achieve advanced road segmentation in urban scenes. The full architecture of our approach is illustrated in Figure 5.5, capturing the synergy between the two modules, *Module A* (SP-CNN) and *Module B* (Semi-Supervised Modified-CycleGAN).
Initially, Module A employs SP-CNN method for initial segmentation. The specifics of this module, rooted in the foundational work of Chapter 3, are summarized in section 5.4.1. It starts with the segmentation of images into superpixels, with similar features to form a structured and efficient input for the specialized CNN. Then, the CNN classifies these superpixels, providing an initial, coarse semantic segmentation of the road, which serves as a foundational layer for further refinement.
Subsequently, section 5.4.2 explores Module B, which builds upon groundwork of Module A. Here, the Semi-Supervised Modified-CycleGAN is introduced to refine the initial segmentation results. This refinement process, inspired by our earlier work with the CRF model as mentioned in Chapter 4, is specifically targeted at superpixels near the predicted road boundaries. This selective approach significantly improves segmentation accuracy, while maintaining computational efficiency. Module B employs the innovative "RGB-L" dataset, an extension of standard RGB channels with additional segmentation labels, to improve the identification and rectification of segmentation inaccuracies. These advancements collectively address the challenges outlined in 5.1, representing a significant leap in our segmentation capabilities. The overarching steps of our proposed method are as follows:

Module A: SP-CNN Steps

1. Segmenting the image into superpixels using SP-CNN method, forming groups of pixels with similar characteristics for more coherent segmentation.

2. Developing comprehensive image descriptors for each superpixel, covering a range of image features.

3. Projecting superpixels onto a regular grid structure, facilitating effective convolutional processing.

4. pixel-wise classification of superpixels, using a tailored CNN for initial road segmentation.

Module B: Semi-Supervised Modified CycleGAN Steps

5. Generating a 4-band imagery dataset, named "RGB-L", by merging specific areas around road boundaries from the original images with their CNN label masks. This process enriches the segmentation data with an additional layer of detail.

6. Segmenting each RGB-L image into overlapping local patches building a new augmented training data set from a single region to balance segmentation accuracy with computational efficiency.

7. Refining segmentation results with our uniquely modified CycleGAN, distinguished from the original CycleGAN by its enhanced objective function and a newly designed generator network with a reduced parameter set. This modification specifically targets improvement of segmentation accuracy along road boundary areas.

8. Merging the segmented local patches produced by the modified-CycleGAN to reconstruct the image at its original scale, thereby creating the comprehensive final segmentation output.

By structuring our approach into these two modules, we offer a detailed yet cohesive view of our methodology, from initial segmentation to refined outputs. The subsequent sections will provide a deeper insight into each module, showcasing their individual contributions and their combined impact on enhancing road segmentation task. The proposed system obtained comparable performance among the top-performing algorithms on the KITTI [Fritsch et al., 2013b] road benchmark and its fast inference makes it particularly suitable for deployment in ADAS.

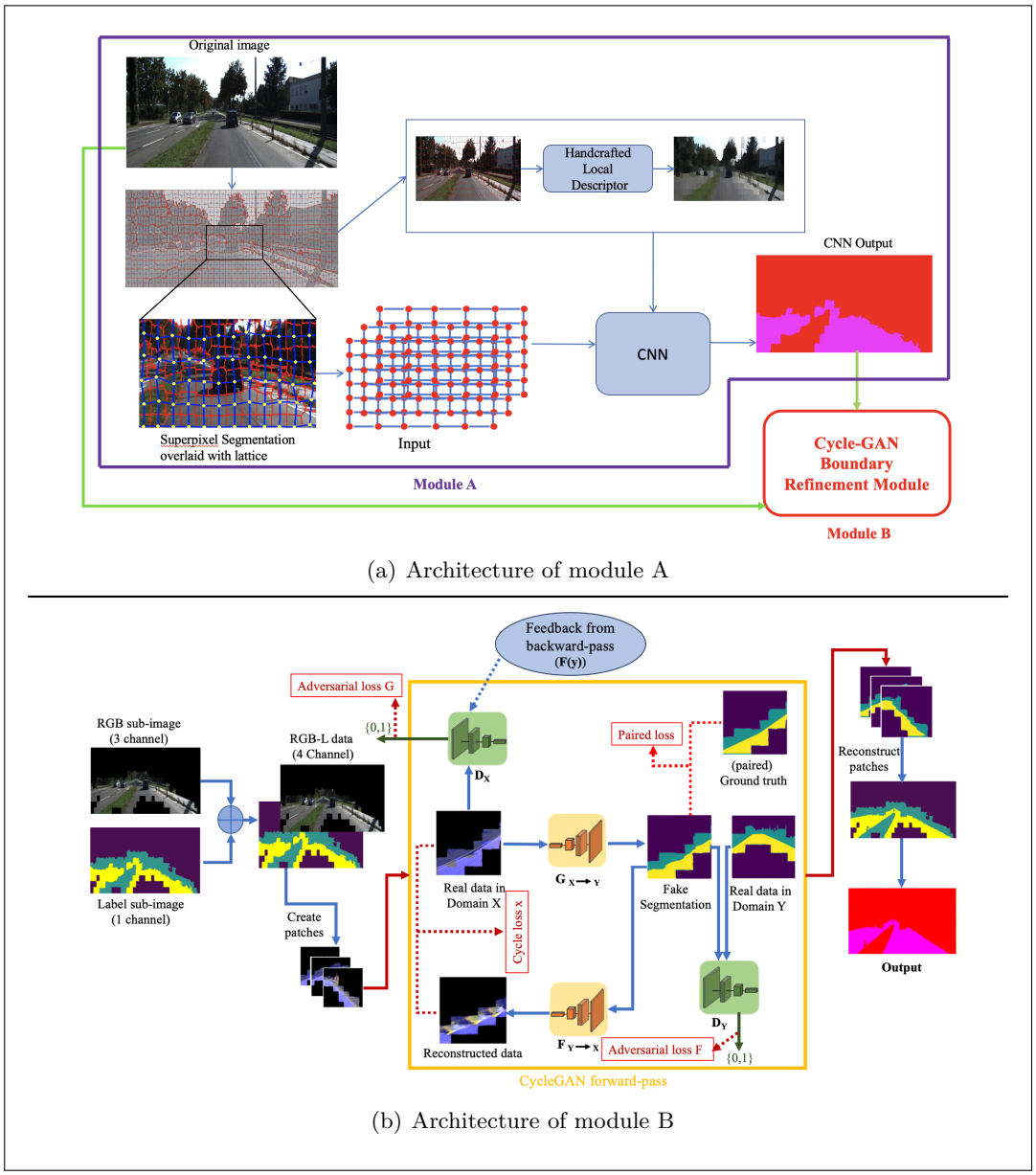


Figure 5.5: Architecture Overview of the Proposed Method. Module A performs initial road segmentation by dividing images into superpixels for CNN feature analysis discussed in chapter 3. Module B then refines this segmentation using a semi-supervised, adapted CycleGAN, focusing on edge accuracy. The forward pass involves an "RGB-L" dataset (RGB images with CNN labels), which is segmented into smaller patches to create an expanded training set. These patches are processed by a modified CycleGAN for precise road boundary prediction, and finally reassembled to produce a detailed and accurate road segmentation.

5.4.1 Condensed Overview of SP-CNN (Module A)

Module A revisits the core elements of our Superpixel-based Convolutional Neural Network (SP-CNN), introduced in Chapter 3 and revisited in 4.4.1. This module plays a pivotal role in the initial phase of our segmentation process, focusing on achieving efficient semantic segmentation of images, even with limited computational resources. Utilizing a modified SLIC algorithm [Achanta et al.,], SP-CNN ensures each superpixel is a coherent assembly of pixels with similar features, creating an optimal input structure for the CNN. These superpixels are then projected onto a mesh grid, facilitating streamlined convolutional processing by the CNN. Additionally, each superpixel is enriched with a 69-dimensional feature descriptor, which includes colour, position, and Local Binary Patterns, significantly boosting segmentation precision.
Our CNN architecture, tailored for this superpixel input, strikes a balance between computational efficiency and effective pixel-wise classification. It comprises a sequence of convolutional and fully convolutional layers, culminating in an initial road segmentation prediction. This initial output is integral for the subsequent refinement process in Module B. In essence, SP-CNN forms the foundation of our segmentation methodology, providing baseline results, that are subsequently enhanced in Module B through the Semi-Supervised Modified-CycleGAN. Figure 5.6 illustrates an example of road segmentation prediction on a KITTI urban scene image, showcasing the effectiveness of our SP-CNN method as developed in Chapter 3.

5.4.2 Segmentation Refinement with modified Cycle-Consistent Adversarial Networks

While the superpixel-based convolutional network, detailed in Chapter 3, has significantly reduced computational demands and obtained a satisfactory accuracy level, challenges in achieving a balance between precision and computational speed persist. Complex road environments, featuring occlusions, ambiguous shadows, and extensive paved surfaces, complicate the task of precise road segmentation. Roads that are not well-maintained, along with areas such as grass or side-walks, are prone to misclassification as part of the road boundaries, as exemplified in figure 5.7. Furthermore, the independent prediction of label variables in CNNs fails to account for the dependencies and relationships among output variables, which are crucial for achieving seamless semantic segmentation.
Although our CRF-based refinement method, introduced in Chapter 4, addresses some of these issues by capturing the inter-variable relationships, it does not always perform optimally due to variations in the distribution of training and real-world testing data. Adequate training data could mitigate this problem, yet the labour-intensive nature of manual annotation presents a significant barrier. In addition, CRF techniques are often constrained by their reliance on specific relational models, which can limit their applicability across diverse scenarios.

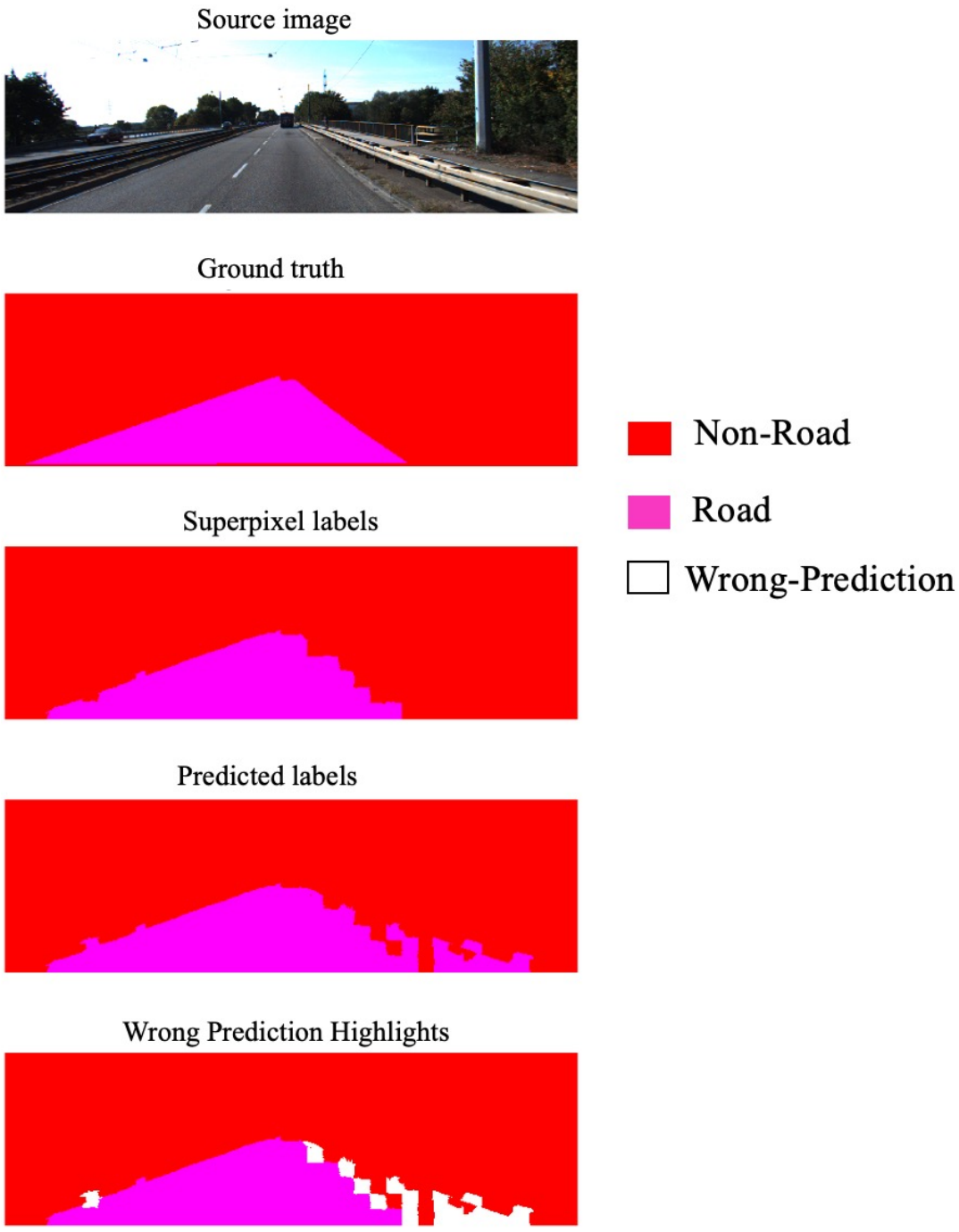


Figure 5.6: Sample result from the KITTI dataset using the Superpixel-CNN method. In the final row, white areas indicate pixels incorrectly predicted by our approach.

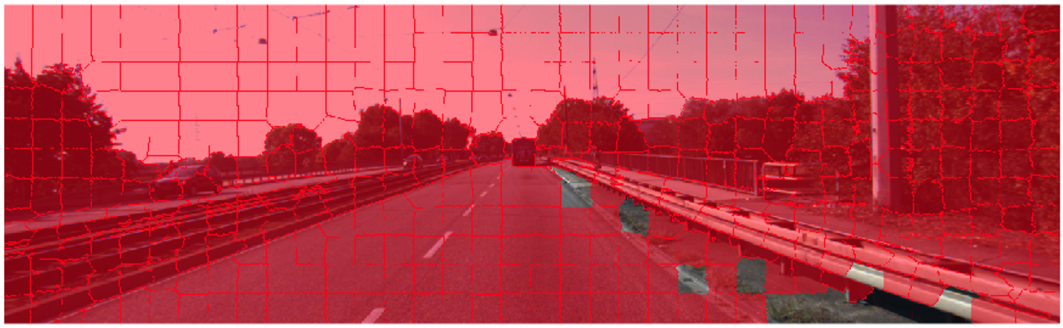


Figure 5.7: Predominant Errors Along Road Boundaries. This example highlights how similar textures between the road and its side result in incorrect predictions, with the road boundary being misidentified as part of the road.

To this end, we introduce a semi-supervised modified CycleGAN with a special emphasis on enhancing road segmentation performance. The original CycleGAN [Zhu et al., 2017], as described in Section 5.3.3, leverages unpaired data from two distinct domains, X and Y , to facilitate image-to-image translation. This translation process is governed by a bidirectional mechanism, the forward pass (X to Y to X) and the backward pass (Y to X to Y), ensuring cycle consistency. Specifically, images from domain X (in our case, "RGB-L" images) are first translated to domain Y using the mapping function $G_{X \to Y}$, and then cycled back to domain X using $F_{Y \to X}$. The backward pass follows a similar cycle, but starting with domain Y. This bidirectional cycle consistency is crucial, especially in the absence of paired data, to maintain the integrity of the translation process.
Our modification to the original CycleGAN architecture is tailored to better suit road border segmentation. Unlike the standard CycleGAN, which typically deals with standard RGB images, our source domain consists of 4-band imagery, combining RGB data with an additional label channel (RGB-L). This integration of label information is aimed at enhancing the translation relevance for segmentation tasks. Additionally, our semi-supervised model introduces a paired L_1 loss function, applied to a subset of our input domain (the RGB-L images) and their corresponding targets in the ground truth database. This modification is intended to improve the precision of fine segmentation, a critical aspect for accurate road delineation. Furthermore, to increase the efficiency of our segmentation task, we have optimized the generator network in the original CycleGAN, reducing its computational demands without compromising performance. These adaptations collectively form our 'modified' CycleGAN. The detailed architecture of the discriminator and generator in our modified CycleGAN, along with the adaptations made, will be elaborated in sections 5.4.2 and

5.4.2. Subsequently, section 5.4.2 will discuss the adapted adversarial loss function, highlighting the nuances of our semi-supervised approach.

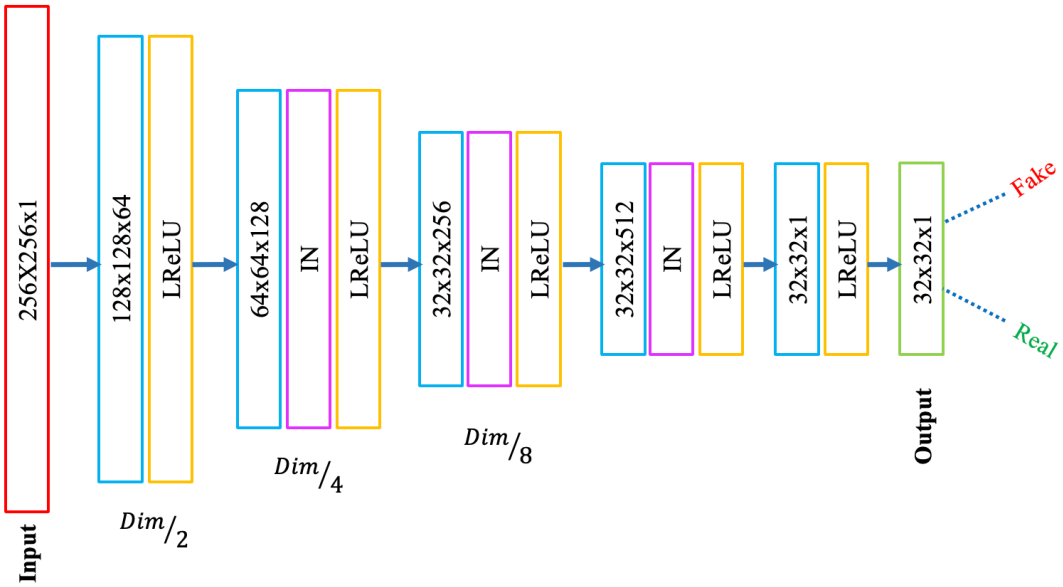


Figure 5.8: The architecture of discriminator D_Y adapted from [Isola et al., 2017, Zhu et al., 2017], consisting of five convolutional layers with a kernel size of 4×4. The first three layers use a stride of $(2, 2)$, successively halve the input's spatial dimensions, resulting in $Dim/2$, $Dim/4$, and $Dim/8$ respectively, where Dim is the dimension of the input image. The final two layers employ a stride of $(1, 1)$, with 'same' padding applied, to maintain the spatial dimensions of the preceding feature maps. This results in an output of size 32×32 from an input of size $256 \times 256 \times 1$, providing a dense prediction that classifies 70x70 patches of the input image as real or fake.

Discriminator Network

Our CycleGAN employs two discriminators, D_X and D_Y, with Figure 5.8 specifically illustrating the structure of D_Y. The discriminators are modelled on the PatchGAN framework [Isola et al., 2017, Zhu et al., 2017], which enhances the quality of generated images by concentrating on individual sections, or "patches", rather than the entire image. This approach allows for a more nuanced discrimination process, where the focus is on the texture and style of localized areas of the image.
In our network, each discriminator outputs a 32×32 matrix, processing overlapping

70×70 input image patches. This results in a detailed map, where each 32×32 element corresponds to a 70×70 receptive field in the input. The PatchGAN discriminator, in this work, comprises a fully convolutional network featuring five layers. The network reduces the spatial dimensions of the input image through the first three layers using a stride of 2. The last two layers use a stride of 1, which, combined with 'same' padding, preserves the dimensions of the feature maps. As a result of this architectural strategy, the spatial resolution of the input is methodically condensed by an overall factor of 8 through the network. The final layer outputs a single-channel 32×32 matrix, with values between 0 and 1, indicating the likelihood of each patch being real or fake.
Each convolutional layer is followed by a "Leaky ReLU" activation function. Unlike the standard ReLU, Leaky ReLU allows a small, non-zero gradient, when the unit is not active. This helps to maintain a gradient flow through the network and ensures, that even neurons that would otherwise be inactive can continue to adapt during training, thus improving the robustness of the network. Following the Leaky ReLU, instance normalization is applied, normalizing the outputs of each feature map independently. This differs from batch normalization, which normalizes across the entire batch of data. In the context of the discriminator, it encourages the model to focus on structural patterns rather than being influenced by contrast or lighting differences, which can vary significantly between real and fake images. Notably, the first convolutional layer does not include instance normalization. This is often a deliberate design choice since instance normalization can remove valuable information about the overall distribution of pixel intensities when applied too early in the network. It is important to retain some of the raw information from the initial features, as these can be crucial for the discriminator to make accurate assessments.
For discriminator D_Y, the input consists of single-channel images, either real images from domain Y (ground truth) or fake images (generated ground truth), having a size of $256 \times 256 \times 1$ according to our dataset. Each output element of 32×32 reflects the authenticity of a corresponding 70×70 patch in the input. D_X, although similar in architecture, differs in that it processes four-channel RGB-L images from our dataset, each of size $256 \times 256 \times 4$. The output size remains the same, ensuring a consistent evaluation across domains. The objective function for each discriminator is designed to measure the accuracy of the network in classifying real and fake patches. For real images, the objective is to align the discriminator output with a matrix of ones, indicating 'real'. Conversely, for synthetic images, the objective aligns with a matrix of zeros, denoting 'fake'. Through this adversarial process, the discriminators become adept at evaluating the veracity of localized image regions, thereby contributing to the overall image synthesis framework.

Generator Network

The generator network in our approach, diverges significantly from the original CycleGAN, which is typically characterized by a encoder-transformer-decoder architecture. In the original CycleGAN, the encoder employs three downsampling blocks (beginning with a stride-1 7×7 convolution, followed by two stride-2 3×3 convolutions, each

coupled with instance normalization and ReLU) to decrease feature map dimensions. The transfer stage comprises nine residual blocks, which process residual features relative to the input at each layer. The decoder, mirroring the encoder, incrementally increases the spatial dimensions of the feature maps through three upsampling blocks, aligning the output scale with the input.
Our modifications to the generator target the enhancement of road segmentation. We have altered the residual blocks, as illustrated in figure 5.9. Unlike the original design, our modified blocks consist of three convolutional layers with dimensions $1 \times 1 \times 128$, $3 \times 3 \times 128$, and $1 \times 1 \times 256$, respectively. Each layer is followed by instance normalization, with the first two also incorporating Rectified Linear Units (ReLU) for activation. The filter count in the first two convolutional layers is halved to 128, compared to the original CycleGAN, and the last layer has 256 filters. Additionally, we've reduced the number of residual blocks from nine to six. These changes lead to a substantial decrease in the overall model complexity, with our generator having $2,038,337$ parameters, a significant reduction from the $11,380,289$ parameters in the original CycleGAN's generator. This optimization reduces computational costs by approximately a factor of 5, while enhancing performance.
In refining our generator, we paid particular attention to the choice of activation functions within the residual blocks. In exploring different activation functions within the generator's residual blocks, we observed that some, particularly Leaky ReLU (LReLU), tended to result in the network learning unintended features related to the mesh grid boundaries used in Module A, rather than the more crucial road boundaries. This often resulted in a coarse, stair-shaped appearance in road boundary predictions. To address this, we opted for Rectified Linear Units (ReLU), which proved more effective in focusing the learning process on the finer, true road boundaries, rather than the superimposed mesh grid. This choice was crucial in reducing the stair-step artifacts and achieving a segmentation that more accurately reflects the actual road contours, as opposed to the stair-step patterns imposed by the mesh grid.

Adversarial Training

This section focuses on the distinct objective function integral to our network, which is an adaptation of the CycleGAN framework, particularly suited for specific image-to-image translation tasks. In the CycleGAN framework, as elaborated in section 5.3.3, a notable innovation is the use of an objective function that combines unique loss components, including full cycle consistency, which proves invaluable for unpaired image-to-image translation tasks. CycleGAN utilizes two generators, $G : X \rightarrow Y$ and $F : Y \rightarrow X$, along with two discriminators, D_Y and D_X. The effectiveness of a generator is measured by its ability to produce images that are indistinguishable from real images by the discriminator. The ideal scenario for a generator is when the discriminator classifies its generated images as real (closer to 1).
We have redefined the total objective function of the original CycleGAN, previously

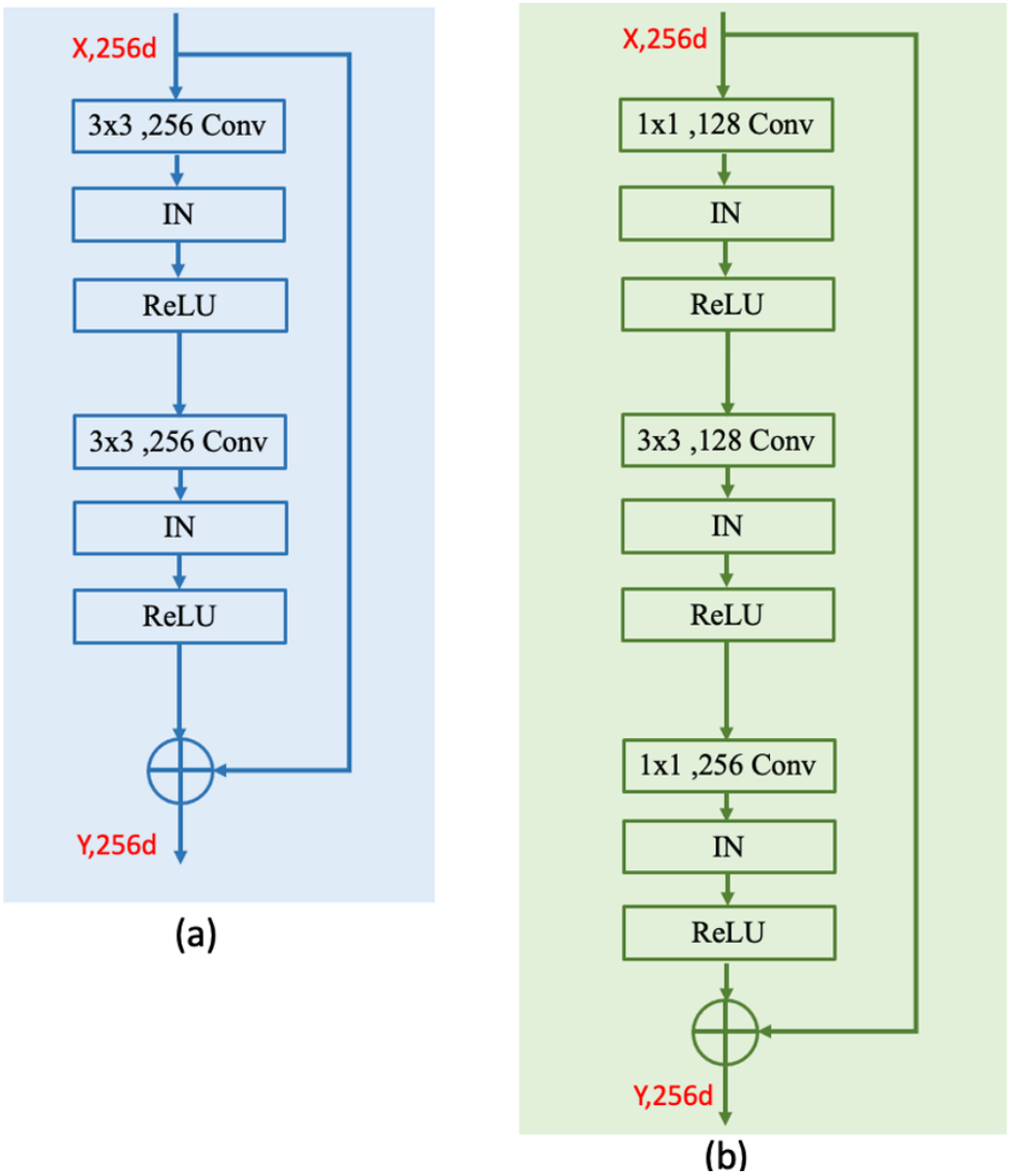


Figure 5.9: The architecture of the residual Blocks in original CycleGAN(a) and our Modified residual Blocks(b)

outlined in equation 5.8, to tailor it more specifically to our needs. Our reformulated objective function is as follows:

$$\begin{aligned} L_{total}(G, F, D_X, D_Y, X, Y) = & L_{adv}^{LS}(G, D_Y, X, Y) + \\ & L_{adv}^{LS}(F, D_X, Y, X) + \\ & \lambda L_{cyc}(G, F) + \\ & \eta L_p(G, F), \end{aligned} \tag{5.10}$$

In this equation, L_{total} represents a composite of several critical loss components. L_{adv}^{LS} is the adversarial loss based on the Least Squares GAN approach, as introduced in [Mao et al., 2017], and is specifically adapted for our generators G and F. The cycle consistency loss, L_{cyc}, ensures coherent image translation between domains, maintaining the essential characteristics of each domain. Furthermore, we introduce L_p, an innovative paired loss designed to exploit the benefits of a subset of paired training data. Detailed formulations of these loss components, which contribute significantly to the overall effectiveness of our model, will be discussed in the following sections. The hyper-parameters λ and η act as balancing factors in this formulation, fine-tuning the influence of the cycle consistency and paired losses, respectively, during the training process. In this setup, X represents our unique 4-channel domain, encompassing RGB data augmented with coarse CNN segmentation results, whereas Y signifies the target domain, characterized by its unpaired ground truth data.

In line with the original CycleGAN methodology, our model employs the Least Squares GAN (LSGAN) approach [Mao et al., 2017] for calculating the adversarial loss in equation 5.6. The LSGAN-based adversarial loss ($L_{\text{adv}}^{\text{LS}}$) functions are defined as follows:

$$\begin{aligned} L_{adv}^{LS}(G, D_Y, X, Y) =& \frac{1}{2} E_{y \sim P_{data}(y)}[(D_Y(y) - 1)^2] \\ &+ \frac{1}{2} E_{x \sim P_{data}(x)}[(D_Y(G(x)) - 0)^2] \end{aligned} \tag{5.11a}$$

$$\begin{aligned} L_{adv}^{LS}(F, D_X, Y, X) =& \frac{1}{2} E_{x \sim P_{data}(x)}[(D_X(x) - 1)^2] \\ &+ \frac{1}{2} E_{y \sim P_{data}(y)}[(D_X(F(y)) - 0)^2] \end{aligned} \tag{5.11b}$$

This choice was motivated by the enhanced stability and effectiveness of training, that LSGAN provides over traditional log-likelihood or cross-entropy loss methods commonly used in GANs. LSGAN methodology, characterized by its focus on minimizing Pearson χ^2 divergence [Mao et al., 2017], offers smoother gradient flows than the log-likelihood or cross-entropy methods. This attribute is crucial for ensuring consistent and effective feedback to the generator, particularly against a strong discriminator. This method avoids excessively penalizing outputs, that are close to the target value, leading to more stable training dynamics and reducing the risk of mode collapse. Such an approach is in harmony with our model's goal of generating high-quality, detailed images.
The respective loss functions for the discriminator D_Y and the generator G within

the LSGAN framework, by using the 0-1 binary coding scheme, are formulated as follows, with analogous calculations applied for D_X and F:

$$\begin{aligned} L_{D_Y} =& \frac{1}{2}\mathbb{E}_{y\sim p_{data}(y)}[(D_Y(y)-1)^2]+ \\ & \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[D_Y(G(x))^2], \end{aligned} \tag{5.12a}$$

$$L_G = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D_Y(G(x))-1)^2], \tag{5.12b}$$

In the 0-1 binary scheme of LSGAN, the value 1 represents the label for 'real' images, while 0 represents 'fake' images. The discriminator loss function, L_{D_Y} for D_Y and analogously L_{D_X} for D_X, aims to assign a score of 1 to real images and 0 to fake images produced by the generators. The generator loss function L_G, specifically for G, and its counterpart L_F for F, are designed to encourage the generators to produce images, that the corresponding discriminators will score as 1, indicating they are indistinguishable from real images.
In traditional GANs, the min-max optimization relies on KL divergence, which measures the deviation of one probability distribution from another. In this setup, the discriminator maximizes the log-likelihood of correctly classifying real and generated (fake) data, while the generator minimizes its likelihood leading to produce data indistinguishable from the real data. In contrast, the LSGAN approach diverges from this paradigm by having both the generator and discriminator work towards minimizing their loss functions. This method emphasizes reducing the squared differences between the real and generated image distributions, streamlining and improving the training process.
The adversarial loss ensures that the generated outputs conform to the appropriate domain characteristics, focusing on domain alignment rather than precise input-generated output pair matching. To address this issue, we introduce our cycle consistency loss, denoted as L_{cyl} and defined by equation 5.7. This loss is calculated using the Mean Absolute Error (MAE) between the reconstructed and original input data, ensuring that domain translations uphold their integrity and fidelity.

A significant adaptation in our model is the inclusion of L_p, a paired loss, which diverges from the $\mathcal{L}_{L_1}$ loss used in the Pix2Pix model (see section 5.3.3). While $\mathcal{L}_{L_1}$ in Pix2Pix is applied over the entire dataset, emphasizing pixel-level accuracy for paired images, our L_p loss is calculated as the L_1 distance between the generated output and the target for only a specific subset of paired training data, denoted as Q. This subset, $Q = \{(x_i^q, y_i^q)\}_{i=1}^{N_Q}$, where N_Q is the number of paired samples in Q, allows us to leverage the precision of paired training on crucial aspects of the model's performance, even with a limited amount of paired data. This strategic use of paired data, albeit in smaller quantities, allows our adapted CycleGAN model to

benefit from the accuracy of paired training, while retaining the flexibility offered by its unpaired data training capabilities. More details will be discussed in section 5.6.

$$L_p(G, F) = \frac{1}{N_Q} \sum_{i=1}^{N_Q} [\|G(x_i^q) - y_i^q\|_1 + \|F(y_i^q) - x_i^q\|_1] , \tag{5.13}$$

In this equation, (x_i^q, y_i^q) represents each paired sample within the Q subset from domains X and Y, respectively. The L_p loss is dedicated to minimizing the L_1 distance for these specific paired samples, ensuring a close correspondence between the generated images and their paired targets within this subset.

Having established the intricate details of our modified CycleGAN architecture and its adversarial training regimen, we now transition to the practical application of this framework in our proposed method. For a deeper understanding of the architectural complexities discussed in this chapter, along with detailed diagrams and explanations, please refer to Appendix II.
The process begins with the generation of patches from the 'RGB-L' imagery. These patches, carefully extracted and enriched with label information, serve as the input to our modified CycleGAN. Here, the enhanced generator and discriminator networks, fine-tuned through our semi-supervised learning approach, work in tandem to refine the segmentation of each patch. Subsequently, the refined patches, now possessing improved segmentation accuracy, especially along the intricate road boundaries, are meticulously reassembled. This reassembly process is a pivotal step, where we ensure that the patches are merged seamlessly to reconstruct the image at its original scale. The result is a comprehensive segmentation output, that not only retains the original image resolution but also exhibits a significantly higher level of detail and precision at the road edges.
In essence, our approach harnesses the power of our proposed SP-CNN architecture to transform initial, coarse segmentations into finely detailed maps. The subsequent sections will delve deeper into the practical implementation of this method, providing a clearer perspective on how each step in our approach contributes to the overarching goal of achieving highly precise and smooth road segmentation.

5.5 Implementation Details

5.5.1 Data Preparation

Our semi-supervised modified CycleGAN method was rigorously tested on the KITTI dataset [Fritsch et al., 2013b], a staple in road segmentation research. Detailed discussions about this dataset and the corresponding evaluation metrics are presented in Section 3.4.1. For robust evaluation across varied urban environments, the KITTI dataset categorizes road scenes into Urban Unmarked (UU), Urban Marked (UM), and Urban Multiple Marked Lanes (UMM), with an additional combined category named 'Urban' that encompasses all three. This diverse categorization ensures a com-

prehensive evaluation of our method across different road conditions. The training set comprises 289 images split across these categories, while the test set contains 290 images, with ground truth labels provided only for the training images. To ensure a balanced evaluation, we divided the training set further into training and validation subsets, respecting the diversity of urban scenarios. Importantly, the images designated for the validation subset were specifically chosen from distinct video sequences, separate from those used in the training subset. This approach ensures that the validation process is robust and representative, free from potential biases that might arise from overlapping or similar content between the training and validation sets.
The consistency in our approach is maintained by adhering to the same data preparation parameters as outlined in our previous works (chapters 3 and 4). This includes using SLIC parameters for superpixel creation and projecting segmented images onto a lattice for CNN processing, ensuring comparability across our methods. For the current approach, we concentrated on the road boundary areas in the KITTI images, by selecting those superpixels, which are touching the road border. The KITTI images vary in size, with widths and heights ranging within $\{1226, 1238, 1241, 1242\}$ and $\{370, 374, 375, 376\}$ pixels, respectively. Based on that, we generated averages around 6 overlapping patches per image, each of size 256×256 pixels with a stride of $s = 196$. This overlap and stride were deliberately carefully chosen to balance comprehensive coverage and computational efficiency, while also aligning seamlessly with the architectural demands of our Modified-CycleGAN network. This sizing ensures compatibility with the input dimensions necessary for both the discriminator, which employs a PatchGAN architecture with a receptive field size of 70×70 and an output dimension of 32×32, and the generator in our network. By selecting these dimensions, we effectively accommodate the intricacies of our discriminator and generator, ensuring that each patch is optimally processed for detailed local analysis as well as comprehensive coverage across the larger image segments.
Each RGB sub-image patch, combined with coarse CNN segmentation data to create a 4-channel input, represents our source domain for the CycleGAN. Concurrently, a similar process generated unpaired target domain patches, representing ground truth, which is derived from the training set. To ensure our focus was on the most relevant segments for road boundary analysis, patches containing less than 2% boundary pixels were excluded. Out of the 1200 samples derived from the original KITTI training set, 1020 were allocated for training. These samples are particularly pivotal in our semi-supervised CycleGAN approach, where the paired loss L_p is applied to a subset of the training data, blending the advantages of both supervised and unsupervised learning. The remaining 180 samples form our validation set, playing a vital role in the model's performance assessment and refinement. Furthermore, 1422 samples were generated from the test set to augment our source domain. The effectiveness of our approach, particularly in smoothing road boundaries, was then benchmarked against state-of-the-art methods and our prior works, using both image perspective and bird's eye view analyses on the KITTI dataset.

5.5.2 Training Details

In training our modified-CycleGAN model, we employed a comprehensive strategy that included the KITTI training set, excluding the validation subset, alongside the KITTI test set. The deliberate addition of 1422 samples from the test set was crucial in diversifying the input scenarios within the source domain. This diversification enriches the model's learning context significantly. By creating this disparity in data volumes (enriching the source domain with test set samples while keeping the target domain limited to training set data), we reinforce the semi-supervised nature of our learning strategy. This approach effectively broadens the model's exposure to various road conditions.
Our training process predominantly involved unpaired mapping between these domains, aligning with the fundamental principles of the CycleGAN framework. To enhance our approach, the paired loss component (L_p), as outlined in equation 5.13, was selectively applied only to the training set samples. This was because these samples uniquely possessed corresponding ground truth labels. This selective use of paired loss in conjunction with the CycleGAN's unsupervised process optimally utilizes the supervised training. This hybrid approach is instrumental in achieving a more precise translation from the source to the target domain, thereby elevating the segmentation accuracy and overall performance of the model.

For the optimization of our modified-CycleGAN model, we employed Mini-batch Stochastic Gradient Descent (SGD) coupled with the Adam optimizer [Kingma and Ba, 2014], setting the momentum parameters to $\beta_1 = 0.5$ and $\beta_2 = 0.999$, with an epsilon of $1E-08$. The batch size was configured to 1, and the initial learning rate was established at 0.0002. The balance parameters λ and η, crucial to our model's performance as outlined in equation(5.10), were set to 10 and 1, respectively (detailed analysis in section 5.6.1). Filter weights were initialized following a Gaussian distribution with a mean of 0.0 and a standard deviation of 0.02, and the data was normalized within the range of $[-1, 1]$. All experiments were conducted using Python on Google Colab with TensorFlow framework (version 2.0). Figure 5.10 illustrates the output from our modified CycleGAN on some various patches.

5.6 Evaluation and Discussion

This section delves into a comprehensive analysis of our model. Initially, we examine various model variants and their empirical outcomes to elucidate the rationale behind our final model's configuration. Key variations explored include a) adjustments in network layering with respect to computational time and accuracy, b) the impact of incorporating the paired loss in the total objective function calculation, c) the effect of varying the number of residual blocks in the Generator network, and d) fine-tuning the penalty values for paired and cycle consistency losses.
Subsequently, we assess the performance of our semi-supervised modified CycleGAN

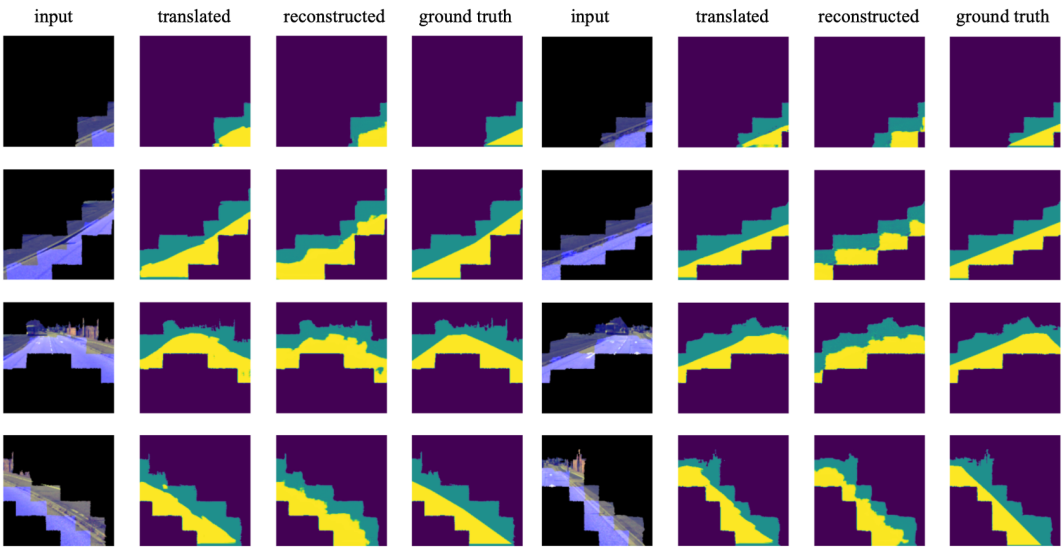


Figure 5.10: Visualization of road segmentation enhancement from a 4-band imagery dataset domain to unpaired ground truth domain, using modified-CycleGAN across eight Sample Patches. 'Input' columns display RGB images overlaid with blue to indicate initial CNN-based coarse segmentation, creating the RGB-L inputs for the source domain. 'Translated' columns depict the unsupervised refinement results after the model translates from RGB-L to the target domain. 'Reconstructed' columns present the cycle consistency enforcement, depicted here without the RGB context for visual clarity. 'Ground Truth' columns are provided for reference, representing the ideal target domain segmentation, noting that these are not directly correlated with the 'Input' images during the model's unsupervised learning process. To assess the model's effectiveness, compare the 'Translated' column to the 'Ground Truth', noting improvements in segmentation accuracy and alignment with the target domain.

(the final model). This evaluation entails a comparison with the pixel grid accuracy from the superpixel-based convolution network described in Chapter 3, our previous work on road segmentation enhancement using the CRF technique (Chapter 4), and the original CycleGAN [Zhu et al., 2017]. In alignment with the KITTI evaluation protocol [Fritsch et al., 2013b], these comparisons are conducted from both an image perspective and a bird's eye view, utilizing the dataset provisions of KITTI.

5.6.1 Model Analysis in Different Scenarios

Our model analysis, performed on the KITTI validation set with a focus on image perspective (referenced in section 5.5.1), was aimed at identifying the optimal configuration for detailed road segmentation. We investigated six different scenarios, de-

Method	**G**enerator network	**T**otal Objective Function	**n**um_residual_blocks	**R**egularisation Parameters
V1	Fig.5.11.(a)	$L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+$ $L_{adv}^{LS}(F,D_X,Y,X)+$ $\lambda L_{cyl}(G,F)$	9	$\lambda = 10$
V2	Fig.5.11.(b)	$L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+$ $L_{adv}^{LS}(F,D_X,Y,X)+$ $\lambda L_{cyl}(G,F)$	6	$\lambda = 10$
V3	Fig.5.11.(c)	$L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+$ $L_{adv}^{LS}(F,D_X,Y,X)+$ $\lambda L_{cyl}(G,F)$	6	$\lambda = 10$
V4	Fig.5.11.(d)	$L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+$ $L_{adv}^{LS}(F,D_X,Y,X)+$ $\lambda L_{cyl}(G,F)$	6	$\lambda = 10$
V5	Fig.5.11.(d)	$L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+$ $L_{adv}^{LS}(F,D_X,Y,X)+$ $\lambda L_{cyl}(G,F)+$ $\eta L_P(G,F)$	6	$\lambda = 10$, $\eta = 5$
V6	**Fig.5.11.(d)**	$\boldsymbol{L_{total}(D,G,X,Y) = L_{adv}^{LS}(G,D_Y,X,Y)+}$ $\boldsymbol{L_{adv}^{LS}(F,D_X,Y,X)+}$ $\boldsymbol{\lambda L_{cyl}(G,F)+}$ $\boldsymbol{\eta L_P(G,F)}$	**6**	$\boldsymbol{\lambda = 10}$, $\boldsymbol{\eta = 1}$

Table 5.1: The Comparison of the different scenarios, which leads to propose our final semi-supervised modified CycleGAN. Method V1 is the Original cycleGAN and the method V6 is our proposed SP-CNN-Modi-CycleGAN

tailed in Table 5.1, ranging from the original CycleGAN (V1), titled **SP-CNN-Orig-CycleGAN**, to our advanced model (V6), titled **SP-CNN-Modi-CycleGAN**. The variations among these scenarios are categorized into four main aspects:

1. *Generator Network Transfer Blocks:* Detailed in the first column of Table 5.1 and illustrated in Figure 5.11, we explored different configurations of the transfer blocks in the generator's encoder-transfer-decoder architecture. This crucial part of our study, aimed at finalizing the design, is also presented in Figure 5.9, where we compare our modified blocks with the original CycleGAN's design. Our focus was on two primary architectures that differ from the original, with the goal of enhancing both segmentation accuracy and processing efficiency. We

also examined various activation functions to improve the network's ability to learn complex patterns.

2. *Objective Function Formulation:* The second column of Table 5.1 describes each scenario's total loss function. All methods, from V1 to V6, incorporate the Least Square GAN (LSGAN) loss (L_{adv}^{LS}) for better learning stability and faster convergence. In our enhanced versions, V5 and V6, we incorporated the proposed paired loss (L_p) into the total objective function.

3. *Residual Blocks Count:* The third column of Table 5.1 specifies the number of residual blocks in each version's generator network. To optimize computational efficiency, we chose to use six residual blocks, in contrast to the original CycleGAN's nine.

4. *Regularization Parameter Weighting:* Displayed in the last column of the table are the varied values for the regularization parameters λ and η within the total loss function. After thorough testing, we established $\lambda = 10$ as the most effective value, following the recommendation in the original CycleGAN paper. Additionally, the paired loss was fine-tuned with different values, finding the most successful implementations in versions V5 and V6.

Our comprehensive analysis has definitively shown that incorporating L_p in versions V5 and V6 greatly enhances our Modified-CycleGAN's performance. This improvement stems from blending supervised learning elements into the CycleGAN's unsupervised framework. By using a subset of the training data with known ground truth, the model effectively integrates the strengths of both learning approaches, resulting in superior segmentation results. Additionally, the unsupervised nature of CycleGAN is particularly advantageous for the KITTI dataset's limited size, ensuring optimal use of available data. The strategic inclusion of L_p complements this approach, leveraging supervised learning to expedite the learning process. This hybrid methodology not only addresses the dataset's constraints but also improves learning efficiency and segmentation accuracy.

5.6.2 Comparative Results Across Model Scenarios

The empirical findings resonate with the theoretical advantages we anticipated. In our quest to identify the model that best balances accuracy with time efficiency, we conducted a series of detailed experiments using image projection on the validation set for each scenario. These experiments, summarized in Table 5.2, provide a comparative evaluation based on the KITTI benchmark metrics. This comparison assesses both performance efficacy and computational demand, the latter gauged by the number of parameters ($\#Params$) in each model's generator network. Additionally, Figure 5.12 visually demonstrates the road segmentation results from the KITTI validation set for each of the six scenarios.
Our analysis of the generator architectures from V3 to V6 reveals a notable improvement in time-efficiency and accuracy over the V1 (original CycleGAN) and V2 models.

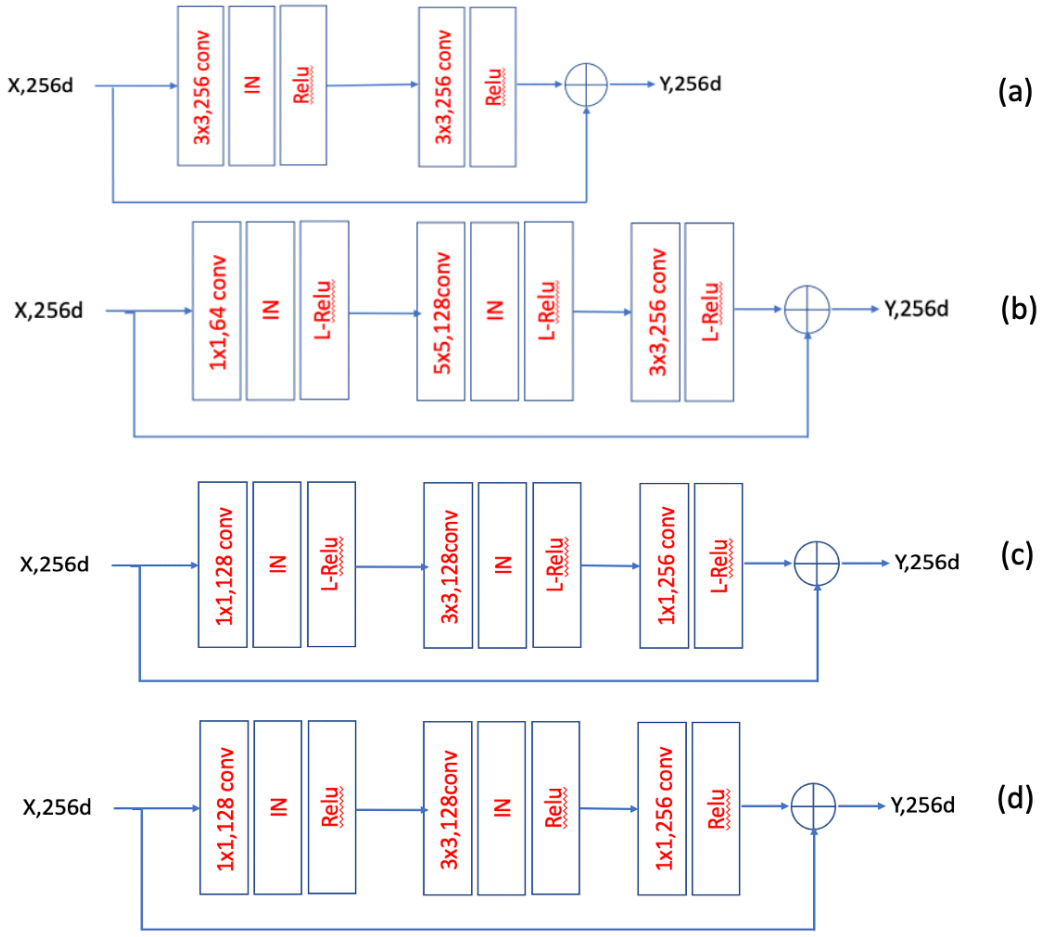


Figure 5.11: Comparative architecture of residual blocks in generator networks across models. (a) Model V1: Original CycleGAN Architecture, (b) Model V2, (c) Model V3, (d) Models V4 to V6, represented as **SP-CNN-Modi-CycleGAN**, showcasing our modified approach.

The use of Leaky ReLU (LReLU) in V2 and V3, while offering a minor gradient benefit for negative inputs, inadvertently led to heightened sensitivity to artifacts. This issue, as highlighted in Figure 5.12 and discussed in section 5.4.2, manifests in the models learning gradients along superpixel boundaries, imposed by mesh grid, rather than the actual road boundaries.

An analysis of the evaluation results from V1 to V4 indicates, that the integration of the L_p loss in V5 and V6 results in marginal but noteworthy improvements in road segmentation performance. This improvement is notably pronounced when dealing with varying levels of complexity within different image distributions, especially when it is applied to the test set rather than the validation set. While it is true that the improvements in accuracy (ACC) and maximum F-measure score(MaxF) in V5 and V6 over V3 and V4 are minimal, these incremental gains are significant in the context

of image segmentation, where even small improvements can be crucial for robust performance. It leads to the small improvements in the road segmentation performance and the robustness of the varying degrees of complexity in the underlying distributions of the different images.
The slight advantage in precision, that V6 offers over V5 and V4 justifies its selection as the final model, particularly for applications where minimizing false positives and ensuring highly accurate road segmentation is crucial. While V5 might be advantageous in scenarios requiring the maximization of true road segment detection (higher recall), potentially at the cost of increased false positives, V6's superior precision makes it more suitable for contexts like autonomous driving, where precision is paramount. This careful consideration led to the choice of V6, detonated **SP-CNN-Modi-CycleGAN**, as the optimal model, prioritizing the precision and reliability of road identification in segmented outputs, as supported by the analysis in Table 5.2 and Figure 5.12.

Method	**A**CC	**M**axF	**P**RE	**R**EC	#Params
V1	95.95%	92.74%	93.41%	92.10%	11.38×10^6
V2	96.02%	92.60%	95.15%	90.47%	3.85×10^6
V3	97.24%	95.15%	94.90%	95.41%	2.04×10^6
V4	97.29%	95.21%	95.20%	95.23%	2.04×10^6
V5	97.31%	95.24%	95.28%	95.20%	2.04×10^6
V6	**97.33%**	**95.26%**	95.58%	94.96%	2.04×10^6

Table 5.2: The comparison of the evaluation results on the KITTI validation set by applying our different proposed scenarios. Method V1 is the Original CycleGAN and the method V6 is our final proposed SP-CNN-Modi-CycleGAN method.

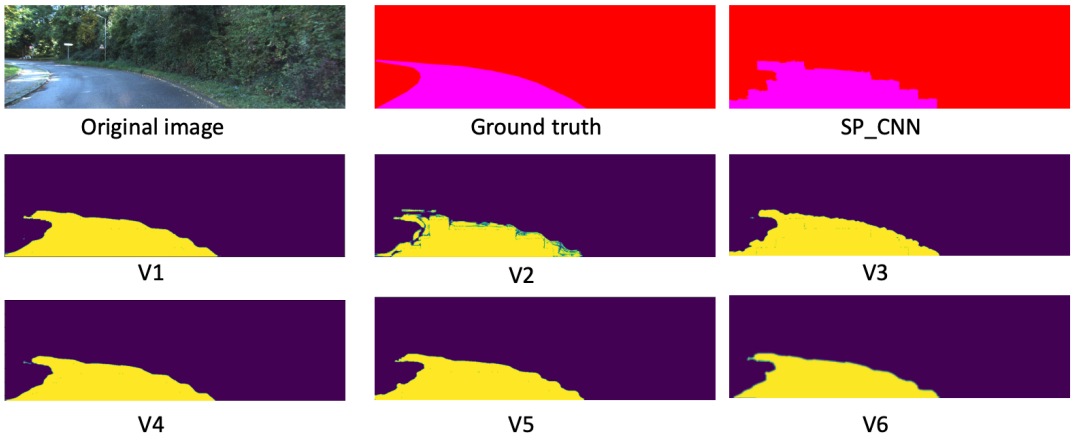


Figure 5.12: Road segmentation results achieved by models V1 to V6

5.6.3 Modified CycleGAN vs. Our Prior Methods on Image Perspective

Following the selection of our V6 model for its precision and computational efficiency, this section transitions to a comparative performance analysis against existing methodologies within the KITTI dataset framework. We evaluate the V6 model's performance, evaluated from an image perspective, with results from our foundational SP-CNN model, our preceding CRF-based enhancement, and the original CycleGAN architecture. The ensuing discussion, detailed in Table 5.3 and visually complemented by Figure 5.13, aims to highlight the significant strides made by our modified CycleGAN in augmenting road segmentation accuracy across diverse urban environments encapsulated in the validation set.

The comparative analysis, using the KITTI validation set, reveals a progressive improvement in segmentation performance from the base SP-CNN method through to the SP-CNN-Modi-CycleGAN approach. The modified CycleGAN method showcased a significant improvement, enhancing accuracy by approximately 3% to an impressive 97.33%, in contrast to the baseline SP-CNN model. This advancement extends approximately 0.5% to 1.5% increase in road segmentation accuracy over the CRF-based method and the original CycleGAN respectively. Notably, the modified CycleGAN method outperforms all previous methods by achieving the highest MaxF score of 95.26%. This score signifies a remarkable balance between precision and recall, indicating a significant enhancement in accurately identifying road segments, while minimizing false positives and false negatives. In terms of precision, it excels with a rate of 95.58%, surpassing all prior methods and emphasizing its ability to reduce false positives. Simultaneously, it achieves a recall rate of 94.96%, indicating its effectiveness in capturing a substantial portion of actual road segments. Illustrated in

Figure 5.13, these results vividly demonstrate the considerable progress achieved with our modified approach, highlighting its efficacy in refining segmentation outcomes.

Method	**A**CC	**M**axF	**P**RE	**R**EC	**F**PR	**F**NR
SP-CNN	94.41%	94.28%	91.41%	97.34%	8.59%	2.60%
SP-CNN-CRF	96.85%	91.47%	92.30%	90.65%	7.70%	2.10%
SP-CNN-Orig-CycleGAN (V1)	95.95%	92.74%	93.41%	92.10%	1.96%	6.48%
SP-CNN-Modi-CycleGAN (V6)	**97.33%**	**95.26%**	95.58%	94.96%	1.43%	8.65%

Table 5.3: Comparative Analysis of Road Segmentation on KITTI Validation Set. This table outlines the performance metrics of various segmentation approaches, including our SP-CNN method, CRF enhancement, the original CycleGAN, and our advanced semi-supervised modified CycleGAN method.

5.6.4 Modified CycleGAN vs. State-of-the-Art Methods on Bird's Eye View

In this section, we delve into the evaluation of our road segmentation methods from the birds-eye perspective. This evaluation involves projecting images onto the ground plane using known camera geometry, with a focus on the KITTI benchmark Test dataset. The results of this assessment are presented in Table 5.4, and our analysis spans across different road types, including UM (Urban Minor), UMM (Urban Minor Major), UU (Urban Unmarked), and URBAN categories. The findings from our evaluation offer valuable insights into the performance of our advanced proposed method, highlighting their respective strengths and areas of improvement.

Our baseline method, SP-CNN, demonstrates promising performance, achieving a maximum F1 (MaxF) of 81.60% for UM_ROAD, showcasing its ability to accurately detect road segments. However, this method exhibits a relatively higher false negative rate (FNR) of 14.60%, indicating instances where road segments were missed.

Building upon SP-CNN, the incorporation of Conditional Random Fields (CRF) in SP-CNN-CRF yields substantial improvements. Notably, it achieves an enhanced MaxF of 83.22% for UM_ROAD, underscoring CRF's potential in mitigating false negatives and enhancing road segmentation.

Our advanced approach, SP-CNN-Modi-CycleGAN, emerges as the most promising method in our evaluation. It attains the highest MaxF across almost all road types, with notable improvements of 85.01% in UM_ROAD and 91.80% in UMM_ROAD.

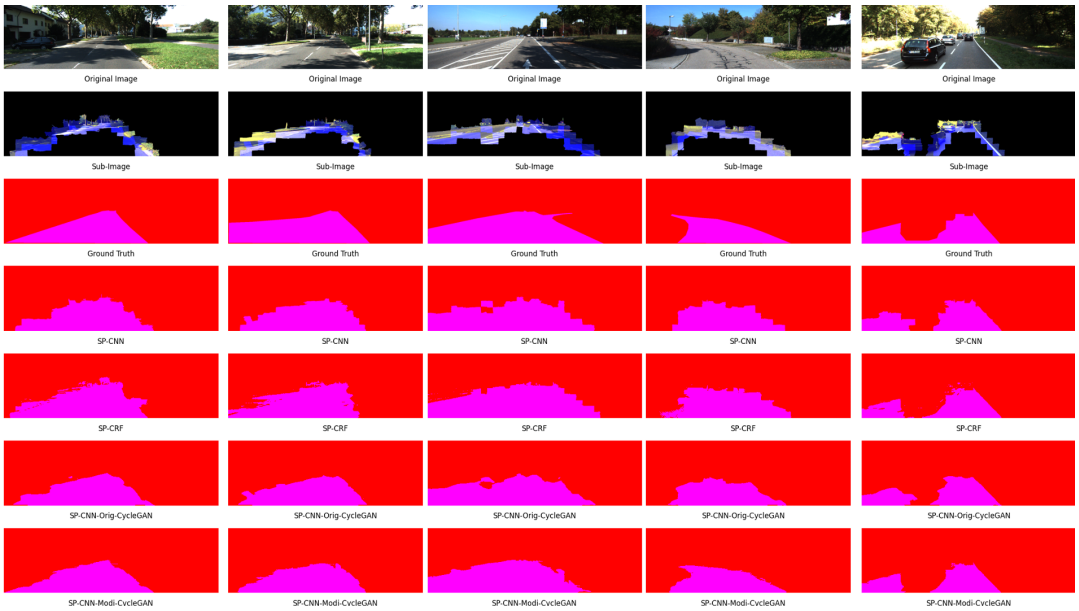


Figure 5.13: Comparative visual analysis of road segmentation results across methods on KITTI validation set. The figure illustrates the original image, focused 'Sub-image' segments on road boundaries alongside with the ground truth and segmented outputs from SP-CNN, SP-CRF, SP-CNN with Original CycleGAN, and the improved SP-CNN with our proposed Modified CycleGAN. . Each method's fidelity to ground truth increases progressively, with SP-CNN-Modi-CycleGAN achieving the closest match.

This signifies the efficacy of our modified CycleGAN approach in accurately identifying road segments and reducing false negatives. Intriguingly, our evaluation reveals a noteworthy aspect in the performance of the SP-CNN-CRF method within the UU_ROAD category, where it marginally outperforms our latest SP-CNN-Modi-CycleGAN approach. This outcome warrants further investigation to understand the distinctive characteristics of UU roads in the KITTI dataset. UU, denoting Urban Unmarked roads, presents a challenge due to the absence of lane markings, particularly in urban environments. The slightly better performance of the SP-CNN-CRF method in the UU_ROAD category could be attributed to CRF's ability to capture contextual information, potentially aiding road boundary delineation in the absence of lane markings. Additionally, factors like road texture distribution, objects, and unique lighting conditions in UU_ROAD scenarios may play a role in these results.
In summary, our comparative evaluation on the KITTI test set reveals significant enhancements in road segmentation accuracy, when employing SP-CNN-Modi-CycleGAN. Compared to the baseline SP-CNN, we observe an approximate 4% improvement in MaxF across all urban categories, indicating the effectiveness of our proposed method. Furthermore, in comparison to our recent CRF-based approach, we achieve an im-

provement of approximately 1%. Particularly noteworthy is the impressive 7% improvement observed in the UMM_ROAD category, relative to the same category in our SP-CNN technique. This suggests that our approach effectively addresses previous inaccuracies attributed to superpixels at the road border. To provide a visual perspective, Figure 5.14 presents a visual comparison of two samples in the birds-eye view on the KITTI test set. This illustration underscores the proficiency of our SP-CNN-Modi-CycleGAN approach in accurately segmenting roads. While the segmentation is generally precise, occasional false detections may occur, often when the algorithm misinterprets shadows covering the street.

Method	Benchmark	MaxF	AP	PRE	REC	FPR	FNR
SP-CNN	UM_ROAD	81.60 %	69.62 %	78.13 %	85.40 %	10.89 %	14.60 %
	UMM_ROAD	85.07 %	79.86 %	85.97 %	84.20 %	15.11 %	15.80 %
	UU_ROAD	78.47 %	65.18 %	74.20 %	83.25 %	9.43 %	16.75%
	URBAN_ROAD	82.36 %	72.31 %	80.48 %	84.33 %	11.27 %	15.67 %
SP-CNN-CRF	UM_ROAD	83.22 %	72.94 %	77.11 %	90.39 %	12.23 %	9.61 %
	UMM_ROAD	90.96 %	84.63 %	87.86 %	94.29 %	14.32 %	5.71 %
	UU_ROAD	**80.02** %	67.93 %	77.56 %	82.64 %	7.79 %	17.36 %
	URBAN_ROAD	85.97 %	77.81 %	82.04 %	90.31 %	10.89 %	9.69 %
SP-CNN-Modi-CycleGAN	UM_ROAD	**85.01 %**	76.86 %	86.98 %	83.13 %	5.67 %	16.87 %
	UMM_ROAD	**91.80 %**	89.25 %	92.94 %	90.70 %	7.58 %	9.30 %
	UU_ROAD	79.49 %	68.66 %	85.19 %	74.51 %	4.22 %	25.49 %
	URBAN_ROAD	**86.90 %**	79.61 %	89.41 %	84.52 %	5.52 %	15.48 %

Table 5.4: Comparative performance on KITTI test set, assessing our proposed SP-CNN, CRF-Enhanced SP-CNN, and modified CycleGAN. The evaluation metrics including Maximum F-measure (MaxF) or F1, Average Precision (AP), Precision (PRE), Recall (REC), False Positive Rate (FPR), and False Negative Rate (FNR). Further details can be found in Section 3.4.1.

5.6.5 Run-time Analysis

This section evaluates the computational efficiency and processing time of our enhanced method, which integrates both coarse segmentation and refinement processes. We compare the efficiency of our modified CycleGAN against the original model by examining the number of parameters ($\#Params$) and floating-point operations ($\#FLOPs$). These metrics help determine which version is more computationally economical and faster. According to the data presented in Table 5.5, computed us-

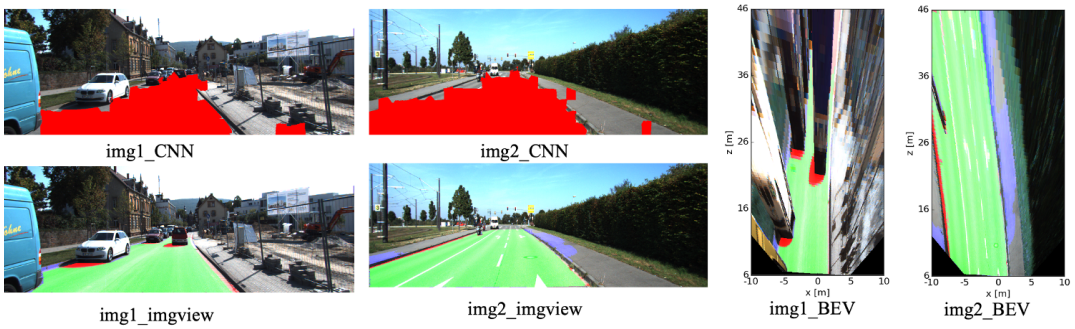


Figure 5.14: Road segmentation results on Bird's Eye View (BEV) from the KITTI Test set, comparing SP-CNN and our Modified CycleGAN approach. False positives are depicted in blue, false negatives in red, and true positives in green.

ing TensorFlow's profiler library (version 2.6), our modifications have significantly optimized the model. By redesigning the generator's residual blocks to be more compact and decreasing their quantity from nine to six, as detailed in section 5.4.2, we've achieved substantial reductions in computational resources. Specifically, our approach demonstrates a decrease in #Params and #FLOPs by factors of approximately 5.58 and 4.42, respectively, when compared to the original CycleGAN model.

Method	#Params	#FLOPs
Original CycleGAN	11.38×10^6	98.85×10^9
Our Modified CycleGAN	2.04×10^6	22.32×10^9

Table 5.5: The Computational cost of our proposed modified CycleGAN method compare to the original CycleGAN

Our comparison extends to evaluating both the accuracy of road segmentation and the average processing time per image against leading semantic segmentation methods. The details are consolidated in Table 5.6, which includes AI_Scores to indicate the computational capabilities of the GPUs used. Our evaluations were conducted on the KITTI UMM test set. Our novel method, SP-CNN-Modi-CycleGAN, leverages superpixels and a streamlined CNN architecture, augmented with a refined cycle-consistent adversarial network. Our refinement method focuses on a carefully chosen subset of pixels adjacent to road boundaries, substantially reducing computational loads. It takes just 0.10 seconds for this process, leading to an overall processing time of merely 0.12 seconds for each image, once superpixel segmentation and CNN processing times are included. This streamlined approach ensures efficiency without

Method	Processor	MaxF	AP	AI-Scores	Runtime(s)
LidCamNet [Caltagirone et al., 2019]	NVIDIA GTX1080 GPUs	96.03 %	93.93 %	17383	0.15
RBNet [Chen and Chen, 2017]	NVIDIA Tesla K20c 5 GB	94.97 %	91.49 %	-	0.18
LODNN [Caltagirone et al., 2017]	NVIDIA GTX980Ti GPU, 6GB	94.07 %	92.03%	16038	0.018
UP_CONV_POLY [Oliveira et al., 2016]	NVIDIA Titan X GPU	93.83 %	90.47	20089	0.083
Ours (SP-CNN) [Zohourian et al., 2018b]	Intel(R) Core(TM) i7-4790K CPU @4GHz	85.07 %	79.86 %	1400	**0.019**
Ours (SP-CNN-Modi-CycleGAN)	NVIDIA T4 GPU	91.80 %	89.25 %	14558	**0.10**

Table 5.6: Comparative analysis of road segmentation performance on the KITTI UMM test set, featuring our SP-CNN and SP-CNN-Modi-CycleGAN methods, against leading state-of-the-art methods. Metrics include Maximum F-score (MaxF) or F1, Average Precision (AP), and execution times, with AI-Scores indicating processor computational capabilities. Only peer-reviewed publications are considered for this comparison. For detailed AI-Scores, visit AI Benchmark.

compromising on precision. This efficiency, achieved on the specified GPU setup, underscores our method's viability for real-time applications, effectively mitigating performance degradation while optimizing processing speed and maintaining a desirable balance between accuracy and time efficiency.

5.7 Conclusion

In this chapter, our objective was to propose an innovative approach aimed at further enhancing the accuracy of semantic segmentation, building upon the foundations laid in the initial work. We desired to advance the capabilities of semantic segmentation by leveraging the potential of a Superpixel-based Convolutional Neural Network. Our overarching goal remained the achievement of real-time performance for a diverse set of semantic segmentation tasks, all while ensuring practical applicability for real-time systems. To demonstrate the practicality and effectiveness of our approach, we focused our efforts on road segmentation, employing the KITTI dataset as the proving ground for our experiments. Our key emphasis was on addressing the limitations posed by challenging conditions, including factors like shadows on the road surface, variations in illumination, and the presence of neighbouring patterns such as sidewalls. These challenges had previously hampered our earlier approaches.
To mitigate these issues, we introduced a novel semi-supervised modified CycleGAN technique aimed at enhancing road segmentation accuracy, even when dealing with limited annotated data. Our method deviates significantly from conventional solutions based on CNNs or CRFs, that rely on large sets of annotated training images or specific high-order potentials to model the pairwise dependencies in segmentation

outcomes. Instead, our proposed semi-supervised modified CycleGAN leverages cycle consistency during the learning process and reduces the burden of human annotation by using only a fraction of annotated images. Notable distinctions from the original CycleGAN include alterations to the generator architecture and the incorporation of a paired loss in the overall objective function. This approach significantly improves road segmentation performance, while reducing computational demands compared to the original CycleGAN. Despite the higher resource requirements of our current GPU-based method compared to earlier CPU-based models, comparative analyses using the KITTI dataset reveal our approach surpasses the Superpixel-based Convolutional Neural Network by $4-7\%$ and achieves competitive results against more sophisticated semantic road segmentation techniques.
In summary, this chapter presents a promising solution for real-time semantic segmentation, applicable to a diverse array of tasks beyond road segmentation. Our proposed smoothness method enhances the performance of our Superpixel-based Convolutional Neural Network, offering a practical solution for real-world applications.

6 Summery and Outlook

In this concluding chapter, we consolidate the principal contributions and discoveries of our research, as detailed in this thesis. Additionally, we engage in a critical discussion of the unresolved issues and propose prospective avenues for forthcoming investigations. This discourse is particularly pertinent given the persistent and substantial hurdles facing real-time automated scene segmentation.

6.1 Summery

Deep learning has pushed the boundaries of almost every vision-related subdomain with the help of parallel computing power (in particular highly parallel GPU architectures) and large-scale datasets. Semantic image segmentation, which is in the core of the field of Computer Vision and scene understanding, has been studied and boosted by the extraordinary ability of Convolutional Neural Networks (CNN), as a kind of deep learning model, to generate high-level, semantic image features. The performance of CNN-based methods has drastically improved with deeper network architectures and more training data. However, the power consumption and sheer size of such models prevent their use in robotic applications. The demand for Computer Vision applications with real-time capabilities has stimulated research interest in the study of efficient and effective object recognition algorithms, where methods are able to provide an output solution with a time budget, which can be run on embedded devices with limited resources and amount of time. In this thesis, we focus on this fundamental task and present efficient methods, both in terms of required time and resources, on the task of semantic image segmentation, which constitute valuable assets for real-time applications such as autonomous driving.
In the scope of this work a CNN-based neural network for semantic segmentation task has been developed. The work was embedded in the context of autonomous driving, where an effective and efficient method for road scene understanding from various complex urban environments is devised. However, the overall approach is transferable to other use cases. The goal was to provide an accurate understanding of the objects in the environment by pixel-wise segmentation approach in an efficient way. Pixel-wise semantic segmentation is a well-known representation for road scene understanding in autonomous driving applications due to its ability in describing a wide variety of scenes. Despite many problems, this main goal has been achieved and the final model has obtained competitive results with advanced methods. To achieve this goal, we have designed fully coarse to fine semantic segmentation pipelines and improved the efficiency and robustness of the entire system in adverse environmental conditions.
In the initial stages of this thesis, specifically in Chapter 2, we embarked on an ex-

tensive exploration of the foundational aspects of deep neural networks, along with a comprehensive review of seminal works in the realm of semantic segmentation. This set the stage for the ensuing chapters (3, 4, 5), where we delved into the core contributions of our research.
Our investigation was structured around three pivotal objectives. First, we aim to develop a semantic segmentation approach that is not only efficient in terms of computational resources but also quick to execute, making it suitable for real-time applications. Second, we focus on improving the segmentation accuracy, transitioning smoothly from coarse to fine-grained analysis. Third, we strive to enhance the system's ability to generalize across different scenarios, particularly those with variations in data distribution not covered by the training dataset, without relying solely on pixel-specific annotations. Our efforts have been directed towards fulfilling these objectives, with a particular focus on system efficiency, thereby contributing to each domain significantly.

6.2 Contributions and Discussion

The main contribution of this thesis was presented in the following three chapters:

Objective 1: Budget-aware method in terms of Time, Resource and computational costs

Chapter 3: Our first contribution in Chapter 3 focused on the learning a supervised semantic segmentation model with memory-efficient and a time budget, which makes it suitable to be used successfully for real-time applications. In particular, this contribution has been developed for the use case of autonomous driving applications like road segmentation. We divided the pipeline of our semantic segmentation approach in three key steps, namely over-segmentation, feature extraction, pixel-wise classification based on CNN.
We constructed an special input data model based on superpixels (for an efficient over-segmentation of the image) and high dimensional feature channels. superpixel creation starts from a grid partition of the image, and then, refines the partitioning by moving the boundaries of the superpixels towards the object boundary to obtain irregular but roughly hegemons segmentations. The core idea is to reduce the computational complexity by segmenting the image into homogeneous regions (superpixels) and feed image descriptors extracted from these regions into a CNN rather than working on the pixel grid directly. Feeding a network with almost well segmented " superpixel" units enables the network to learn local information like contrast, shape, texture, etc. much better rather than using raw image pixels. In addition, spatial relationship information is preferentially preserved in the proposed method due to the use of superpixels rather than patch-wise model or pooling methods.
For the feature extraction we extracted manually a group of efficient features that enables us to model better relevant object characteristics in each superpixels and speed up the computations. We tested different combinations of features and decided on a

particular combination, which gives the best performance.
To enable the necessary convolution operations on the irregular arranged superpixels, we introduce a lattice projection scheme as part of the superpixel creation method, which composes neighbourhood relations and forces the topology to stay fixed during the segmentation process. Reducing the input to the superpixel domain allows the CNN's structure to stay small and efficient to compute while keeping the advantage of convolutional layers. This results in significant reduction of the computational costs for a densely labelled map prediction. The method is generic and can be easily generalized for segmentation tasks other than road segmentation. The proposed method comprises the following steps:

- First, segmenting the image into superpixels, wherein the superpixels are coherent image regions comprising a plurality of pixels having similar image features.
- Then determining image descriptors for the superpixels, wherein each image descriptor comprises a plurality of image features.
- The superpixels are assigned to corresponding positions of a regular grid structure extending across the image.
- This lattice together with the image descriptors are fed to the convolutional network based on the assignment to classify the superpixels of the image according to semantic categories.

Experiments conducted on two widely-used challenging datasets with different resolution sized of images comprising urban scenarios, i.e., the KITTI [Fritsch et al., 2013a] and the Cityscapes [Cordts et al., 2016] datasets. The results show that our proposed method greatly improves segmentation efficiency with very limited computational resources, while achieving accuracy comparable to expensive state-of-the-art methods.

Objective 2: A refinement strategy to compensate the trade-off between accuracy and efficiency

Chapter 4: In our second contribution towards an efficient solution to the semantic segmentation task, we improved the performance of the previously proposed method by injecting spatial context into our segmentation results obtained from our superpixel-based semantic segmentation convolutional neural network(CNN). In previous chapter, we discussed our memory-efficient and fast semantic segmentation model (SP-CNN) able to run in CPU at acceptable frame-rates with consumption very low computation resources. Despite our great achievement to obtain low computational time in both training and testing phases, some problems such as lower resolution of the superpixel domain, variation in brightness and appearance apart from occlusion, and inability of CNNs to capture the global context information caused naturally lower accuracy compared to high cost state of the art methods. In other

words, label prediction by CNN-based method is done for each local neighbouring interdependently without including the information from surrounding labels, whereas the labels produced by CNN at nearby pixels are correlated, due to the overlap of their receptive fields. Such dependencies are not explicitly modelled and produce coarse outputs.
The increase in segmentation performance from previous chapter, achieved by utilizing a probabilistic graphical model called Conditional Random Field (CRF) [Lafferty et al., 2001] as post-processing step to model various pixel label dependencies and refine weak and coarse segmentation outputs. The total framework comprised two aspects for coupling local and global evidences. We combined the local image classification information extracted from CNN part with global information of neighbouring pixels obtained from CRF to decide accurate pixel label. Experiments show that without using of global classification, the segmentation performs poorly, especially in inhomogeneous superpixels. However, with a hypothetical solution, the segmentation outperforms the CNN results and achieves comparable segmentation accuracy with other state of the arts methods. Reducing the input to the superpixel domain allows the CNN's structure to stay small and efficient to compute, while keeping the advantage of convolutional layers and makes them eligible for ADAS. Employing CRF mitigates the balance between accuracy and computational expense. The proposed system obtained comparable performance among the top-performing algorithms on the publicly available road benchmarks and it's fast inference makes it particularly suitable for real-time applications.

Objective 3: Enhance system adaptability to unseen Data and Lower annotation requirements through unsupervised domain translation

Chapter 5: In this chapter, we delve into our pioneering approach within the realm of unsupervised semantic segmentation, marking a significant stride towards achieving Objective 3. Our approach leverages a sophisticated strategy of unpaired domain translation to significantly improve the system's adaptability, enabling it to capture the underlying distribution of varied data domains. This capability allows our model to generalize effectively to new, unseen data without the stringent requirement for extensive, pixel-level annotations that strong supervision demands. In the preceding chapters, we introduced methods for rapid and efficient semantic segmentation that are grounded in strong supervision, necessitating a substantial corpus of well-balanced, pixel-annotated training images. While such detailed annotations can push the boundaries of network performance, they come at a high cost in terms of both financial resources and human efforts. Moreover, the prevalent benchmarks in Computer Vision typically adhere to a closed set evaluation framework, assuming that the training and test datasets are drawn from identical distributions. This assumption, however, does not hold in many practical scenarios, such as in robotics and autonomous driving, where the environment is dynamic and unpredictable. The presence of data, that is either not represented in the training set or is unevenly distributed can significantly reduce the learning process. Consequently, the availability

of a diverse dataset and the development of a model capable of adeptly adjusting to novel and unseen data become crucial for enhancing performance, while simultaneously minimizing the reliance on extensive annotation.
In this chapter, we introduce a semi-supervised post-processing approach that leverages the principles of unpaired image-to-image translation to enhance our model's generalizability, while keeping its efficiency high for embedding in real time systems. Our technique employs a tailored version of the Cycle Generative Adversarial Network (CycleGAN), which relies on a limited set of annotated images to dramatically decrease the need for extensive manual labelling. This approach is particularly beneficial in fields, where acquiring annotated data is challenging, such as in robotics and the medical domain. Our experimental framework focuses on urban street scenes, aiming to refine road segmentation under various difficult conditions, including shadows, diverse surface textures like unpaved areas, and areas where the road surface closely resembles adjacent patterns, such as side-walks. The method encompasses the following key contributions:

1. We introduce a modified cycle consistence generative adversarial network to enhance the semantic segmentations result ,obtained from our first proposed approach, in semi-supervised learning.

2. The proposed adversarial method enforce cycle consistency to learn the mapping between unpaired 4-D channel images and the label (ground truth) domain. These 4-D channel images comprise the original RGB layers augmented with a fourth channel, that represents the segmented areas derived from our superpixel-based CNN semantic segmentation technique. By processing these images through our enhanced CycleGAN model, we are able to extract detailed and fine-grained segmentation results.

3. The computational cost is reduced in two ways. First by redesigning the residual blocks of the original CycleGAN into a shorten structure and reducing their number of parameters to keep the computational effort low. Second, the adversarial learning procedure is limited to the road boundary for boosting the segmentation performance.

4. Contrary to the original CycleGAN, we used a semi(un)paired dataset and we performed segmentation enhancement by applying L1 loss between the output and target. Consequently, the enhancement quality improves better than the original CycleGAN and previous methods.

6.3 Outlook

We believe that the following research avenues can use the works developed in this thesis:

1. To better understand the implications of these research, the work presented in this thesis can be further extended to estimate the semantic segmentation into several classes. It only requires an annotated superpixel dataset for different classes and investigating the appropriate initial number for the grid partition of the image based on the size of the images in the respective dataset.

2. The work was embedded in the context of autonomous driving, where an effective and efficient method for road scene understanding from various complex urban environments is devised. However, the end goal is to obtain the efficient and robust system capable of fine-grained pixel-wise segmentation with the high level of generalizability for any unseen but similar images and reduced human and resource costs for embedding in real time systems. This paradigm of less labels or computational effort is basically suitable respectively in those domains where image collection is cumbersome, such as the medical field or robot navigation systems, where in-time responding is critical.

3. The simplicity of implementation, integrating in real-time systems, and generalizability are very challenging problems for autonomous driving or any real-time applications. In this thesis we explored different supervised,unsupervised and semi supervised scenarios for fine-grained semantic segmentation task. We combined the advantages of a kind of classical clustering-based image segmentation technique with supervised CNN network to obtain pixel-wise classification label and later on explored unsupervised probabilistic graphical model and advanced unsupervised domain translation methods to enhance the segmentation results. Due to the comprehensive study and results explored in this thesis, expensive cost of pixel-level annotations, and the lack of generalizability of the fully supervised machine learning methods, we believe that the future works will still concentrate on lowering the annotation and computational cost using semi-supervised or fully unsupervised systems through domain translation techniques, and improving the performance by focusing on optimization parameters such as decaying the weight of cycle consistency loss as training progresses, or weighting cycle consistency loss by the quality of generated images.

4. Building upon above foundation, a promising avenue for future research emerges from the integration of Conditional Random Fields (CRF) with Cycle Generative Adversarial Networks (CycleGAN) in semantic segmentation. Integrating CRF with CycleGAN leverages CRF's ability to model spatial consistency and enforce smoothness in segmented regions, enhancing the integrity and coherence of segmentation outcomes. This synergy is academically significant as it combines CycleGAN's strength in adapting models across domains without paired examples with CRF's capacity to maintain spatial relationships and reduce noise

by considering the contextual interactions between pixels or segments. Such an integration also facilitates more effective learning from limited annotations by leveraging unlabelled or weakly labelled data, exploiting structural information to make the model more robust to variations and ambiguities.

5. Although our semi-supervised method (proposed in chapter 5) achieved satisfactory results, there still remain inherent drawbacks of such methods. First, the trade-off between prediction accuracy and cost-effectiveness remains challenging. The lower accuracy may induce from either wrong-predicted superpixel (apart from those along the road border) or saturation of CNN-model learning ability due to the small-sized of annotated dataset. The higher computational effort comes from unsupervised generative adversarial model complexity. Second, in general, the training data in the supervised CNN-based model strongly affects the stability of the segmentation model, and a large number of well-balanced and high-quality labelled images are required to have a stable and high-performance system. With this concept in mind, we will focus on the optimal network architecture for semantic segmentation as a future perspective. Inspired by the recent advance in unsupervised domain translation, it would be interesting to investigate a complete end-to-end semantic segmentation strategy based on our superpixel CNN-based model, that is capable of segmentation and refinement in one pipeline within a time budget.

List of Figures

List of Tables

Math-Symbols

W	Weight vector in an ANN, comprising learnable weights $[w_1, \ldots, w_n]$, associated with a single neuron, containing the weights for each input to that neuron.	11
b	Bias term in an ANN, a scalar added to the input's weighted sum to adjust the neuron's output	11
f	Activation function in an ANN, introducing non-linearity to determine the neuron's output relevance	11
z	Result of a neuron's linear transformation in an ANN, serving as the input to the activation function	12
a	Output of a neuron in an ANN post-activation, indicating the neuron's contribution to the network's output	12
B	The series of bias vectors in the neural network, where $b^{[l]}$ represents the bias vector for the l-th layer	17
$\mathrm{n}^{[l]}$	The number of neurons in layer l	17
$\mathrm{W}^{[l]}$	The weight matrix associated with the layers l and $l-1$	17
$\mathrm{b}^{[l]}$	The vector of biases associated with the layer l	18

$a^{[l]}$	The vector of activations of the neurons in the layer l	18
$z^{[l]}$	The vector of the sum of the weighted output of the neurons in the layer l	18
$f^{[l]}$	The activation function applied to the the neurons in the layer l	18
$\mathcal{C}$	The loss(cost) function	18
η	The learning rate	19
∂	The partial derivative	19
$\delta_j^{[l]}$	The error in the j^{th} neuron in the l^{th} layer	20
ν	The variance of a distribution, quantifying the dispersion or spread of its values around the mean	22
n_{in}	The number of neurons in the preceding layer	22
n_{out}	The number of neurons in the subsequent layer	22
σ_{He}	The standard deviation for "He" initialization method, used to initialize weights effectively for layers with ReLU activation functions	22
L_1	Lasso regularization term, adding the absolute values of the weights as a penalty to the loss function to promote sparsity and prevent over-fitting by driving many coefficients to zero	23
L_2	Ridge regularization term, adding the square of the weights as a penalty to the loss function to prevent over-fitting by encouraging a distribution of smaller weights	23
λ	Regularization parameter controlling the impact of the regularization terms L_1 and L_2 on the loss function, balancing the trade-off between fitting the training data and maintaining a simple model	23

$\in$	A mathematical notation denoting membership, indicating that an element belongs to a set or a collection	23
R	The set of real numbers, indicating that the regularization parameter λ can take any real value	23
$\hat{z}_j^{[l]}$	The normalized version of the aggregated weighted inputs $z_j^{[l]}$ for the j-th neuron in layer l, ready for the activation function	24
$\mu_j^{[l]}$	The empirical mean of the aggregated weighted inputs $z_j^{[l]}$ for the j-th neuron in layer l, computed across a batch of M samples	24
$\nu_j^{[l]}$	The empirical variance of the aggregated weighted inputs $z_j^{[l]}$ for the j-th neuron in layer l, computed across a batch of M samples	24
ϵ	A small constant added to ensure numerical stability	24
$z_{\text{norm}j}^{[l](i)}$	The intermediate normalized version of the aggregated weighted inputs $z_j^{[l]}$ for the i-th sample and j-th neuron in layer l, before scaling and shifting	24
γ	A learnable scale factor in batch normalization, allowing the network to adjust the normalization effect	24
β	A learnable shift factor in batch normalization, allowing the network to adjust the normalization effect	24
L_{CE}	Cross-Entropy loss	25
N	The total number of samples in the dataset or the batch over which the loss function is computed	25
$\log(\cdot)$	Natural logarithm function	25
L_{MSE}	Mean Squared Error loss	26

Softmax	A function used mostley as an activation function, that converts logits or raw scores from the neural network's output layer into probabilities, ensuring the output probabilities sum to 1	26
e	Euler's number, a mathematical constant approximately equal to 2.71828, used as the base for the exponential function	26
$\mathrm{L}_{\mathrm{Softmax}}$	Softmax loss or categorical cross-entropy loss	26
F_{h}	The height of a filter or kernel in a convolutional layer	29
F_{w}	The width of a filter or kernel in a convolutional layer	29
$z_{i,j}^{[l]}$	The output at position (i, j) in the feature map of the l-th convolutional layer, computed before applying the activation function	29
$\mathbf{X_{ij}}$	The input data or multi-dimensional feature matrix	29
$\mathbf{W}^{[l]}$	The weight matrix of a filter in the l-th convolutional layer	29
H_{fm}	The height of a feature map in a Convolutional Neural Network	30
W_{fm}	The width of a feature map in a Convolutional Neural Network	30
$\mathrm{H}_{\mathrm{out}}$	The height of the output feature map produced by a transposed convolution operation	32
$\mathrm{W}_{\mathrm{out}}$	The width of the output feature map produced by a transposed convolution operation	32
s	The stride of the transposed convolution operation, indicating the step size for moving the filter across the input feature map	32
N_{px}	The total number of pixels in the image	49

N_{sp}	The desired number of superpixels	49
S	The interval between superpixel centers on a grid, calculated as $\sqrt{N/K}$, which approximates the size of a superpixel	50
$Cent_i$	The 5D vector representing the cluster center of the i-th superpixel, consisting of CIELAB colour space values l_i, a_i, b_i and spatial coordinates xx_i, yy_i	50
(x_p, y_p)	The coordinates of the pixel	50
m_{slic}	A compactness parameter in SLIC superpixel segmentation that controls the balance between colour similarity and spatial proximity	50
d_{color}	The Euclidean distance in CIELAB colour space between a pixel and a superpixel center	50
$d_{Spatial}$	The Euclidean distance in the spatial domain between a pixel and a superpixel center	50
D_s	The combined distance measure for SLIC superpixel segmentation, incorporating both colour and spatial distances, with spatial distances normalized by the grid interval S	50
(P, R)	In the context of the extended Local Binary Patterns (LBP) operator, (P, R) defines the neighbourhood around a central pixel, where P is the number of sampling points and R is the radius from the central pixel to the sampling points	54
τ	The threshold function used in the Local Binary Patterns (LBP) operator	54
$P(Y\|X)$	Probability distribution of the label configuration Y given the input data X, typically represented within the framework of a Conditional Random Field (CRF) using a Gibbs distribution.	88

$E(Y\|X)$	Gibbs energy of a labeling Y given the input data X	90
C_G	Set of cliques in the graph $G = (V, E)$, where a clique is a subset of nodes in the graph that are completely connected.	90
$Z(X)$	Normalization factor in the Gibbs distribution of a CRF, ensuring that the probabilities sum up to one over all possible label configurations.	90
$\varphi(x_i, y_i)$	Unary potential function for node i in a Conditional Random Field (CRF), representing the cost or likelihood of node i taking a specific label based on the observed data	90
ψ_{ij}	Pairwise potential function between nodes i and j in a Conditional Random Field (CRF), representing the interaction or dependency between these nodes	91
$Y^{(t)}$	The label configuration for all pixels in the image at iteration t in an iterative optimization algorithm, where t denotes the iteration number	96
MaxItr	Maximum number of iterations	96
T	The energy threshold specified as a termination criterion in an iterative optimization algorithm	96
ΔE	The minimum change in energy required between iterations to continue in an iterative optimization algorithm	96
$\Delta E_i^{(t)}$	The potential energy change for pixel i at iteration t, when considering a new label in an iterative optimization algorithm	96

$\bar{k}$	An auxiliary vector in GMM, that assigns each pixel to one of the K GMM components, indicating the specific component, that best represents the pixel's characteristics.	97
k_i	The GMM component assigned to the i-th pixel, based on whether the pixel is associated with the road or non-road class in the Adaptive ICM algorithm	97
μ	The mean vector of a GMM component, typically representing the average RGB value for the pixels associated with that component.	97
Σ	The covariance matrix parameter of a Gaussian Mixture Model component, capturing the variance and correlation of pixel intensities	97
π	The weighting coefficient in a Gaussian Mixture Model, determining the mixing proportion of each component in the model	97
$\bar{\theta}$	Parameters of the Gaussian Mixture Model within the ICM algorithm, including means, covariances, and mixing coefficients	97
k_i^*	The optimal GMM component assignment for the i-th pixel in GMM	98
$\bar{\theta}^*$	The optimized set of GMM parameters (μ, Σ, and π) after learning in the Adaptive ICM algorithm	98
$\mathcal{N}$	Denotes the Gaussian (or normal) distribution function, characterized by a mean (μ) and covariance (Σ), used in the context of Gaussian Mixture Models to represent the components of the mixture	99
$m_{i \to j}(y_j)$	Message from node i to node j regarding state(label) y_j in the LBP method	102

$b_i(y_i)$	The belief at node i regarding its potential label y_i in Loopy Belief Propagation, calculated based on incoming messages from neighboring nodes,except the one from which it is currently sending a message to	102
O	It is a mathematical notation, that describes the algorithm's complexity, helping to quantify performance as the input size grows.	105
ω	A parameter in the pairwise potential function, that penalizes label discrepancies between adjacent pixels, promoting label consistency in adjacent regions	107
σ_α	A parameter in the pairwise potential function of a Conditional Random Field (CRF), related to the appearance kernel, controlling the influence of spatial distance in the mean-field approximation	109
σ_β	A parameter in the pairwise potential function of a Conditional Random Field (CRF), related to the appearance kernel, controlling the influence of colour similarity in the mean-field approximation	109
σ_γ	A parameter in the pairwise potential function of a Conditional Random Field (CRF), related to the smoothness kernel, controlling the emphasis on spatial proximity in the mean-field approximation	109
G	Generator function in CycleGAN	127
D	Discriminator function in CycleGAN	127
θ_g	The learnable parameters of the generator in a Generative Adversarial Network	127

$G(z)$	Output of the generator G in a GAN, where z represents a random input noise vector drawn from a distribution	127
$p(z)$	The distribution from which the random input noise vector z is drawn in a Generative Adversarial Network	127
θ_d	The learnable parameters of the discriminator in a Generative Adversarial Network	127
$p_{data}(y)$	The real data distribution from which real data samples y are drawn in a Generative Adversarial Network	128
$D(y)$	The probability estimate output by the discriminator D for real data samples y	128
$D(G(z))$	The probability estimate output by the discriminator D for synthetic data samples generated by $G(z)$	128
G^*	The optimal generator in a Generative Adversarial Network, achieved when the adversarial training process converges, producing synthetic data indistinguishable from real data by the discriminator	128
D^*	The optimal discriminator in a Generative Adversarial Network, achieved when the adversarial training process converges, perfectly distinguishing between real and synthetic data	128
$\mathbb{E}$	The expectation operator, used in the context of Generative Adversarial Networks to calculate the expected value over a probability distribution	128

$V(G, D)$	Value function in GANs defining the adversarial game, where G seeks to minimize this function and D aims to maximize it, based on the expectation over real data distribution and synthetic data from G.	128
$G(z\|x)$	The output of the generator in a Conditional Generative Adversarial Network, where the generation process is conditioned on additional information x	130
$D(y\|x)$	The output of the discriminator in a Conditional Generative Adversarial Network, evaluating the authenticity of data y conditioned on additional information x	130
$\mathcal{L}_{L_1}$	The L1 loss function, ensuring pixel-level similarity between the generated and target images, thus preserving content accuracy in the image translation process	133
$\|\cdot\|_1$	A mathematical notation used to denote the sum of absolute values of the elements in a vector or the absolute differences between corresponding elements in two vectors	133
$G_{X\to Y}$	The mapping function in a CycleGAN framework that translates images from domain X to domain Y	135
$F_{Y\to X}$	The mapping function in a CycleGAN framework, that translates images from domain Y back to domain X, ensuring cycle consistency in the translation process	135
D_Y	In CycleGAN, D_Y is the discriminator tasked with distinguishing between real images from domain Y and fake images generated by the generator G translating images from domain X to domain Y	135

D_X	In CycleGAN, D_X is the discriminator responsible for differentiating between real images from domain X and fake images produced by the generator F translating images from domain Y to domain X	135
$L_{adv}(G, D_Y, X, Y)$	The adversarial loss for the generator G in CycleGAN, calculated with respect to the discriminator D_Y, ensuring generated images from domain X are indistinguishable in domain Y	136
$L_{adv}(F, D_X, Y, X)$	The adversarial loss for the generator F in CycleGAN, calculated with respect to the discriminator D_X, ensuring generated images from domain Y are indistinguishable in domain X	136
$L_{cyl}(G, F)$	The cycle consistency loss in CycleGAN, ensuring that an image translated from one domain to another and back again closely resembles the original image, preserving content integrity across translations	136
$L_{\text{adv}}^{\text{LS}}$	Least Squares GAN (LSGAN) adversarial loss	147
L_G	Loss function for the generator G	148
L_F	Loss function for the generator F	148
L_p	Paired loss introduced in the Modiefied-CycleGAN model to leverage a subset of paired training data, enhancing the model's performance by focusing on the L1 distance between the generated output and the target within this subset	148
Q	A specific subset of the training data consisting of paired samples, where each pair (x_i^q, y_i^q) is used to calculate the paired loss L_{paired} in the Modiefied-CycleGAN model	148

N_Q	The number of paired samples in the subset Q, used in the calculation of the paired loss L_{paired} in the Modiefied-CycleGAN model	148
(x_i^q, y_i^q)	A paired sample in the subset Q used in the Modiefied-CycleGAN model, where x_i^q is from domain X and y_i^q is its corresponding target image from domain Y	149

Acronyms

$2D$	Two-dimensional
$3D$	Three-dimensional
$4-D$	Four-Channel
ACC	Accuracy
$ADAS$	Advanced Driver-Assistance Systems
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
BEV	Bird's Eye View
BP	Back Propagation
BP	Belief Propagation
CE	Cross-Entropy
CG	Computer Graphics
$cGAN$	Conditional Generative Adversarial Network
CIE	International Commission on Illumination
$CIELAB$	CIE L*a*b* Color Space

$CMYK$	Cyan, Magenta, Yellow, Key/Black
CNN	Convolutional Neural Network
$COCO$	Common Objects in Context
CPU	Central Processing Unit
CRF	Conditional Random Field
$CycleGAN$	Cycle Consistence Adversarial Network
$DCNN$	Deep Convolutional Neural Network
DDN	Deep Deconvolutional Networks
DNN	Deep Neural Network
EM	Expectation Maximization
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
FPV	First Person View
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GT	Ground Truth
HOG	Histogram of Oriented Gradients
HSV	Hue, Saturation, Value
ICM	Iterated Conditional Mode

Acronyms

IoU	Intersection-over-Union
$KITTI$	Karlsruhe Institute of Technology and Toyota Technological Institute
Lab	lightness, a(green / magenta), b (blue / yellow)
LBP	Local Binary Patterns
LBP	Loopy Belief Propagation
$LIDAR$	Light Detection and Ranging
$LReLU$	Leaky Rectified Linear Units
MAE	Mean Absolute Error
MAP	Maximum A Posterior
$MaxF$	Maximum F-measurement
ML	Machine Learning
MRF	Markov Random Field
MSE	Mean Squared Error
$PASCAL$	Pattern Analysis, Statistical Modelling and Computational Learning
$Pix2Pix$	Pixel to Pixel
PRE	Precision
REC	Recall
$ReLU$	Rectified Linear Units
RGB	Red, Green, Blue
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent

$SIFT$	Scale-Invariant Feature Transform
$SP-CNN$	Superpixel-based Convoloutional Neural Network
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UM	Urban Markings
UMM	Urban Multiple Marked
UU	Unmarked Urban
VOC	Visual Object Classes

Glossary

Accuracy Is a metric that describes how the model performs across all classes..

Activation Functions Non-linear transformation functions within ANNs that define decision boundaries by setting thresholds, crucial for enabling ANNs to solve complex, non-linear problems.

Adversarial Loss A loss function used in training GANs, that quantifies how well the generator can deceive the discriminator into believing its outputs are real, encouraging the generator to produce increasingly realistic images.

Adversarial Training A training methodology where models, typically in a GAN framework, learn to generate data by competing against an adversarial model that tries to distinguish generated data from real data, enhancing the generative model's ability to produce realistic outputs.

AI-Scores A performance metric designed to evaluate and compare the efficacy of different hardware systems in executing deep learning tasks, with higher scores indicating superior performance in terms of computational speed, efficiency, and the ability to process complex AI algorithms.

Artificial Neural Networks (ANNs) Computational models inspired by the human brain, designed to process relationships between data sets, consisting of neurons, weights, biases, and activation functions.

Average Precision (AP) A measure used in object detection to quantify the precision of predictions across different recall levels, effectively summarizing the precision-

recall curve into a single value by computing the average precision values for recall levels over the interval [0, 1].

Backpropagation A method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data is processed, essential for the network's learning process.

Batch Normalization A technique to improve the performance and stability of artificial neural networks by normalizing the inputs of each layer so that they have a mean output activation of zero and a standard deviation of one.

Belief Propagation An algorithm for performing inference on graphical models, such as Bayesian networks and Markov random fields, by passing messages between nodes to compute marginal distributions, thereby facilitating decision-making based on probabilities.

Binary Classifier A type of classification algorithm, that divides instances into two groups or classes.

Bird's Eye View A perspective from above, akin to observing from a bird's high vantage point, offering a comprehensive overview of a scene or area. This viewpoint is particularly useful for creating detailed layouts, maps, or architectural designs, providing a clear, top-down perspective that facilitates spatial understanding and planning..

Complexity (O) In computational theory, complexity, denoted as O, refers to the Big O notation that describes the upper bound of an algorithm's running time or space requirement in terms of the size of the input.

Computational Efficiency A measure of the computational resources required to perform a task, with higher efficiency indicating lesser resource use for the same task, critical in real-time systems.

Conditional GAN (cGAN) A variant of the GAN, where both the generator and discriminator are conditioned on some additional information such as class labels

or data from other modalities, allowing the model to generate images that are conditioned on this information, leading to more controlled and diverse generation processes.

Conditional Random Field (CRF) A Probabilistic model for segmenting and labeling data, utilized in semantic segmentation to consider the spatial relationship between pixels for refined output..

Convolutional Layers The building blocks of CNNs that apply convolution operations to the input, capturing features like edges and textures in the image.

Convolutional Neural Networks (CNNs) A class of deep neural networks, particularly effective in analyzing visual imagery, characterized by their use of convolutional layers to automatically and adaptively learn spatial hierarchies of features.

Cycle Consistency Loss In the context of CycleGAN, a loss function that ensures the original input image can be reconstructed after a round-trip translation (source to target and back to source domain), helping to preserve key attributes across translations.

CycleGAN A Generative Adversarial Network variant for image-to-image translation in unsupervised learning, without requiring paired examples between domains.

Data Augmentation A technique for increasing the amount of training data by applying various transformations to the existing data, such as rotation, scaling, and flipping, to create new and diverse examples.

Deconvolutional Layers Layers that perform the inverse of the convolution operation, often used in CNNs to upsample feature maps and learn dense representations in tasks like semantic segmentation.

Deep Learning A subset of machine learning involving neural networks with many layers, enabling the automatic learning of complex patterns in data for tasks like image and speech recognition.

Dense Conditional Random Field (Dense CRF) An extension of the traditional CRF model, that considers a fully connected graph where every pixel in the image is connected to every other pixel, significantly improving the ability to model fine details and capture long-range dependencies in image segmentation tasks.

Discriminator Networks In the context of Generative Adversarial Networks (GANs), a discriminator network is a model that learns to distinguish between real data (from the dataset) and fake data (generated by the generator network). Its goal is to accurately classify the inputs as real or fake.

Domain Adaptation The task of adapting a model trained on one domain (source) to effectively perform on a different, but related domain (target), especially important in situations where labeled data in the target domain is limited or unavailable.

Dropout Layers A regularization technique where randomly selected neurons are ignored during training, helping to prevent overfitting by making the network's architecture less sensitive to the weights of individual neurons.

Effectiveness Is the ability to produce a better result, one that delivers more value or achieves a better outcome..

Efficiency Is the ability to produce an intended result in the way that results in the least waste of time, effort, and resources.

Energy Function In the context of CRFs and other graphical models, an energy function quantifies the compatibility of a particular labeling of the image with the observed data and prior knowledge, where lower energy values indicate more probable label configurations.

Expectation Maximization (EM) A statistical algorithm used to find maximum likelihood estimates of parameters in probabilistic models with latent variables, by iteratively applying expectation (E) and maximization (M) steps.

F1-Score Is the weighted average of Precision and Recall. F1 is usually more useful than accuracy, especially if you have an uneven class distribution. .

Feature Extraction The technique of identifying and selecting significant features from raw data, crucial for the performance of machine learning applications.

Feature Maps Outputs from the convolutional layers in a CNN, representing the features detected in the input images, such as edges or textures.

Fine-Grained Segmentation A detailed level of segmentation, that aims to distinguish between very similar sub-categories or instances within a broader class, providing more nuanced and precise segmentation results.

Fine-Tuning A process in machine learning where a pre-trained model is adapted to a new, but similar task by continuing the training process. This involves making minor adjustments to the weights of an already trained network, which can significantly improve performance on the new task with relatively little additional data.

Floating Point Operations (FLOP) In computational terms, FLOP refers to the number of Floating Point Operations performed by an algorithm or a model. In the context of deep learning, it's a measure of the computational complexity and efficiency, often used to evaluate the performance and resource requirements of neural network models during training and inference.

Forward Propagation The process in neural networks where input data is passed through the network layers to generate an output. This phase involves the calculation of the weighted sum of inputs, followed by the application of an activation function at each layer.

Fully Connected Layers Layers in a neural network where all neurons from the previous layer are connected to each neuron, often used toward the end of CNNs to flatten the output and make predictions.

Fully Convolutional Networks (FCN) A type of neural network architecture where all layers are convolutional and capable of processing input images of any size to produce correspondingly-sized output maps. This is particularly useful in tasks like semantic segmentation where the goal is to classify each pixel of the image.

Gaussian Mixture Models (GMM) A probabilistic model, that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters, widely used for clustering applications, density estimation, and as a component in more complex models.

Generative Adversarial Networks (GANs) A framework for estimating generative models via an adversarial process, involving a system of two neural networks—generator and discriminator—that contest with each other.

Generator Networks In GANs, the generator network is a model, that learns to generate data that is similar to some real data. It tries to fool the discriminator network into classifying its generated outputs as real.

Gibbs distribution Gibbs distribution representing the conditional probability of a specific label configuration given the input data and model parameters in CRFs, used in energy-based segmentation methods.

Gibbs Energy In the context of graphical models, Gibbs energy is a function that represents the total energy of a system configuration, used to model the probability distribution of states in Markov random fields and CRFs.

Global Information In the context of image segmentation, refers to the high-level, holistic features and relationships within an image, that help inform more accurate and contextually aware segmentation decisions, as opposed to local information derived from immediate pixel neighborhoods.

Gradient Descent An optimization algorithm used for minimizing the cost function in machine learning and deep learning algorithms.

Graph Cut Optimization An optimization technique used in image segmentation, that models the segmentation problem as a graph and finds the minimum cut that separates the graph into disjoint subsets, effectively partitioning the image into object and background segments.

High-Dimensional Feature Channels Feature channels derived from image data that encapsulate detailed information across multiple dimensions, such as color, texture, and spatial location, to enhance the performance of machine learning models.

Higher-Order CRF An extension of the Conditional Random Field framework, that includes potentials over larger sets of variables, not just pairs, allowing for more complex interactions and dependencies to be modeled in tasks like image segmentation.

Hyperparameters Settings or configurations external to the model, that govern the training process, such as learning rate, number of epochs, and architecture design choices. Unlike model parameters, hyperparameters are not learned from the data but are set prior to training to guide the learning process.

Image Descriptors Quantitative representations of basic characteristics within an image, such as edges, textures, or colors, used to describe and identify regions of interest in image processing and analysis.

Image Perspective (Egocentric View) A photographic or visual perspective, which reflects the scene as observed from the viewpoint of the camera or the person capturing the images..

Image-to-Image Translation A process in computer vision where the goal is to learn the mapping between an input image and an output image, often used for tasks such as photo enhancement, colorization, and style transfer.

Inference The process of deducing unknown properties of a probabilistic model given observed data, often involving calculating marginal distributions, expectations, or finding the most probable state of the model.

Instance Segmentation An advanced form of segmentation that identifies and differentiates between individual objects of the same class within an image.

Intersection over Union (IoU) A metric used to evaluate the accuracy of an object detector on a particular dataset. It measures the overlap between the predicted bounding box and the ground truth bounding box.

Iterated Conditional Modes (ICM) A deterministic algorithm used to find a local maximum of the posterior distribution in Markov random fields, by iteratively updating each variable to the mode of its conditional distribution, often used for image segmentation tasks.

L_1 Loss L_1 loss, also known as Mean Absolute Error (MAE), is a metric used in regression and machine learning tasks, that calculates the mean of the absolute differences between the target values and the predicted values. It is robust to outliers as it does not square the errors, making it useful in applications where it is important to treat all deviations from the target equally, regardless of their direction..

L_2 Loss L_2 loss, also known as Mean Squared Error (MSE), is a commonly used metric in regression and machine learning, that calculates the mean of the squares of the differences between the target values and the predicted values. This approach penalizes larger errors more heavily than smaller ones, which can lead to smoother predictions and is sensitive to outliers due to the squaring of the errors.

Label Consistency A desired property in segmentation tasks, where similar regions or adjacent pixels within an image are assigned the same or coherent labels, contributing to the overall accuracy and visual coherence of the segmentation result.

Latent Space In the context of GANs and other generative models, the latent space represents a high-dimensional space of encoded representations from which the generator creates new data instances. Points in this space can be thought of as compressed representations of the data, and interpolating between points in this space can lead to the generation of new data instances.

Lattice Projection A method to organize superpixels into a regular grid structure, facilitating the application of convolutional operations on the otherwise irregularly shaped superpixels.

Least Squares GAN (LSGAN) Least Squares GAN (LSGAN) is a variant of the Generative Adversarial Network, that uses the least squares instead of log-likelihood loss function for the discriminator. This modification leads to minimizing the Pearson χ^2 divergence, resulting in the generation of higher quality images by stabilizing the training process and addressing the vanishing gradients problem found in standard GANs..

Local Binary Patterns (LBP) A feature descriptor used in computer vision and image processing for texture classification. LBP operates by comparing each pixel with its surrounding neighborhood and encoding this relation into a binary code, effectively capturing local texture information.

Loopy Belief Propagation (LBP) An inference algorithm for graphical models that estimates node marginal distributions by passing messages in a graph with loops, used in CRFs.

Loss Functions Functions that measure how well a machine learning model performs, indicating the difference between the model's predictions and the actual data.

Markov Random Fields (MRF) A type of undirected probabilistic graphical model, that represents the joint distribution of variables with a graph, where nodes represent variables and edges represent dependencies, with the Markov property implying that a variable is conditionally independent of others given its neigh-

bors, widely applied in spatial data analysis and image processing to model local interactions.

Max-Product Message Passing A variant of the message passing algorithm, that focuses on the most probable outcomes by maximizing the product of incoming messages and potentials.

Maximum A Posteriori Probability (MAP) A principle in Bayesian inference used to estimate an unknown quantity as the mode of its posterior distribution, effectively combining observed data with prior knowledge to find the most probable value of the unknown parameter given the data.

Mean-Field Approximation A computational technique used to simplify complex probabilistic models by approximating them with a simpler model, where each part of the system is assumed to interact only with an "average" effect of the rest, often used in the context of variational inference to make the problem of estimating probability distributions more tractable.

Message Passing A technique used in algorithms for inference in graphical models, where nodes (variables) exchange information (messages) to compute marginal distributions or perform other inference tasks.

Min-Sum Message Passing A variation of the max-product algorithm, where operations are performed in the log domain, converting products into sums and maximization into minimization.

Mode Collapse A common issue in GAN training, where the generator learns to produce a limited variety of outputs, or even the same output, regardless of the input noise. This is undesirable as it indicates that the generator is not capturing the full diversity of the data distribution.

Modified CycleGAN A variation of the standard CycleGAN architecture, adapted to enhance performance for specific tasks such as road segmentation, often involving adjustments to the network's structure, loss functions, or training regime.

Multi-Layer Perceptrons (MLP) A class of feedforward artificial neural network that consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. MLP utilizes a supervised learning technique called backpropagation for training.

Number of Parameters (#param) In the context of neural networks, the number of parameters (#param) refers to the total count of trainable weights and biases within the model. This metric is indicative of the model's capacity, complexity, and potential for overfitting or underfitting, depending on the size and diversity of the training data.

Object Detection A computer vision technique for locating instances of objects within images or videos. It involves identifying and outlining objects in an image with a bounding box and labeling them.

Optimization Algorithms Algorithms used to adjust the parameters of a neural network to minimize the loss function. Common examples include Stochastic Gradient Descent (SGD), which updates parameters for each training example, and Adam, an algorithm that computes adaptive learning rates for each parameter.

Optimization Strategy (CRF) A methodological approach within Conditional Random Fields aimed at finding the most probable labeling of image regions that minimizes a given energy function, balancing accuracy and computational efficiency.

Overfitting A modeling error in machine learning that occurs when a function is too closely aligned to a limited set of data points, affecting its performance on new data.

Pairwise Potentials Components of the energy function in CRFs that define the cost associated with assigning a pair of neighboring pixels or regions to particular

labels, used to enforce spatial consistency and smoothness in the segmentation output.

Patch In image processing and computer vision, a patch refers to a small, contiguous block of pixels within an image. Patches are used to analyze local features, textures, or patterns.

PatchGAN PatchGAN refers to a discriminator architecture used in certain GAN models, that classifies patches of an image as real or fake, rather than the entire image or each pixerl. This approach focuses on the structure at the scale of these patches and is effective for tasks like image-to-image translation, allowing the model to capture and preserve high-frequency details..

Pearson χ^2 Divergence Pearson χ^2 divergence is a statistical measure used to quantify the difference between two probability distributions, emphasizing the squared difference between observed and expected data. In the context of GANs, minimizing this divergence helps in improving the fidelity of the generated data to the real data distribution.

Pooling Layers Layers within a CNN that reduce the spatial dimensions (width and height) of the input volume for the next convolutional layer, helping to decrease computation and control overfitting.

Precision Is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples predicted as Positive (either correctly or incorrectly)..

Real-Time Applications Applications that require immediate processing and response to input data, where any significant delay would result in a failure to meet the application objectives, such as in advanced driver-assistance systems.

Recall Is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect Positive samples. The higher the recall, the more positive samples detected..

Recurrent Neural Networks (RNNs) A type of neural network where connections between nodes form a directed graph along a temporal sequence, allowing it to exhibit temporal dynamic behavior and use internal state memory to process sequences of inputs.

Refinement Procedures Techniques applied to initial segmentation results to improve their quality, often involving the optimization of label assignments based on additional criteria such as spatial consistency, achieved through methods like CRFs.

Regularization A technique used to prevent overfitting in machine learning models by adding a penalty on the complexity of the model, often through a regularization term in the loss function.

Reinforcement Learning An area of machine learning concerned with how agents ought to take actions in an environment to maximize the notion of cumulative reward.

ReLU (Rectified Linear Unit) A type of activation function that is defined as the positive part of its argument. Where an input is positive, ReLU outputs the same value; where it is negative, ReLU outputs zero. It introduces non-linearity in the neural networks and helps them to learn complex patterns.

RGB-L A dataset used in semi-supervised CycleGAN for road segmentation, consisting of RGB images augmented with CNN-derived labels. This dataset facilitates precise boundary prediction and segmentation enhancement by providing both color information and pre-segmentation cues.

Semantic Categories Classifications within an image that correspond to meaningful categories in the context of the scene, such as roads, vehicles, and pedestrians, used in semantic segmentation tasks.

Semantic Segmentation The process of partitioning a digital image into distinct segments, making the image's representation simpler and more meaningful for analysis.

Semantic Segmentation Refinement The process of enhancing the accuracy of initial semantic segmentation predictions, often by incorporating additional post-processing techniques such as Conditional Random Fields (CRFs) to incorporate spatial context and improve the delineation of object boundaries.

Semi-Supervised Learning A learning approach that involves a small amount of labeled data and a large amount of unlabeled data during training. It combines supervised and unsupervised learning techniques to improve learning accuracy.

Simple Linear Iterative Clustering (SLIC) An efficient algorithm for superpixel segmentation that adapts k-means clustering to group pixels into superpixels based on their color similarity and proximity in the image space.

Softmax A mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the exponentials of the input numbers. Often used in the output layer of a classifier to provide a probabilistic interpretation of class membership.

Spatial Context Information related to the arrangement and relationship of objects within an image. In semantic segmentation, incorporating spatial context helps in achieving more coherent and accurate segmentation results by considering the interactions between neighboring pixels or regions.

Superpixel Segmentation The process of partitioning an image into clusters of pixels, called superpixels, which are more perceptually meaningful and homogeneous than individual pixels, often used to reduce computational complexity in image analysis.

Superpixels Clusters of pixels in an image based on their color and spatial proximity, used to decrease image complexity for more efficient analysis.

Supervised Learning A machine learning approach where the model is trained on a labeled dataset, which includes both input data and the corresponding correct outputs.

Support Vector Machine (SVM) A powerful and versatile supervised learning algorithm used for classification and regression tasks, which constructs a hyperplane or set of hyperplanes in a high-dimensional space to separate different classes with as wide a margin as possible.

Ternary Classifier A classification model that categorizes instances into one of three possible classes.

Unary Potentials In CRFs, unary potentials represent the cost of assigning a specific label to an individual pixel or superpixel, based solely on the observed data at that location, without considering the labels of neighboring pixels.

Underfitting A scenario in machine learning where a model cannot capture the underlying trend of the data. Underfitting would occur if the model is too simple to understand the complex structure of the data.

Unpaired Image Translation A process in image processing, where the goal is to learn a mapping between two image domains without requiring corresponding images in each domain, allowing for the transformation of images from one style or set of characteristics to another in the absence of direct pairings.

Unsupervised Learning A type of machine learning that looks for previously undetected patterns in a dataset with no pre-existing labels and with minimal human supervision.

Urban Scene Segmentation The process of dividing an urban environment image into semantically meaningful parts, such as roads, buildings, and vehicles, often using techniques like CNNs for applications like autonomous driving.

Vanishing or Exploding Gradient Problems that occur when training deep neural networks, where gradients can become too small (vanish) or too large (explode), making the network hard to train and affecting the convergence of the model.

Bibliography

[Achanta et al.,] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. SLIC superpixels. Technical report, EPFL Technical Report 149300.

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

[Aggarwal et al., 2018] Aggarwal, C. C. et al. (2018). Neural networks and deep learning. *Springer*, 10:978–3.

[Alvarez and Ĺopez, 2010] Alvarez, J. M. Á. and Ĺopez, A. M. (2010). Road detection based on illuminant invariance. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):184–193.

[Amza, 2012] Amza, C. (2012). A review on neural network-based image segmentation techniques. *De Montfort University, Mechanical and Manufacturing Engg., The Gateway Leicester, LE1 9BH, United Kingdom*, 1:23.

[Arnab et al., 2015] Arnab, A., Jayasumana, S., Zheng, S., and Torr, P. H. (2015). Higher order potentials in end-to-end trainable conditional random fields. *ArXiv Preprint ArXiv:1511.08119*, 2.

[Arnab et al., 2018] Arnab, A., Zheng, S., Jayasumana, S., Romera-Paredes, B., Larsson, M., Kirillov, A., Savchynskyy, B., Rother, C., Kahl, F., and Torr, P. H. (2018). Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35(1):37–52.

[Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *ArXiv Preprint ArXiv:1511.00561*.

[Badrinarayanan et al., 2017] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

[Bali and Singh, 2015] Bali, A. and Singh, S. N. (2015). A review on the strategies and techniques of image segmentation. In *2015 Fifth International Conference on Advanced Computing and Communication Technologies*, pages 113–120. IEEE.

[Bangar, 2022] Bangar, S. (Jul 5, 2022). ResNet Architecture Explained. https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d.

[Besag, 1986] Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302.

[Beucher and Lantuejoul, 1979] Beucher, S. and Lantuejoul, C. (1979). Use of watersheds in contour detection. Int. In *Workshop on Image Processing, CCETT/IRISA, Rennes, France*.

[Bishop and Nasrabadi, 2006] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

[Burghouts and Geusebroek, 2009] Burghouts, G. J. and Geusebroek, J.-M. (2009). Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113(1):48–62.

[Caesar et al., 2018] Caesar, H., Uijlings, J., and Ferrari, V. (2018). Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1209–1218.

[Caltagirone et al., 2019] Caltagirone, L., Bellone, M., Svensson, L., and Wahde, M. (2019). LIDAR–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131.

[Caltagirone et al., 2017] Caltagirone, L., Scheidegger, S., Svensson, L., and Wahde, M. (2017). Fast LIDAR-based road detection using fully convolutional neural networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1019–1024. IEEE.

[Camilus and Govindan, 2012] Camilus, K. S. and Govindan, V. (2012). A review on graph based segmentation. *International Journal of Image, Graphics and Signal Processing*, 4(5):1.

[Chen et al., 2014] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected CRFs. *ArXiv Preprint ArXiv:1412.7062*.

[Chen et al., 2016] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *ArXiv Preprint ArXiv:1606.00915*.

[Chen et al., 2017] Chen, Y.-H., Chen, W.-Y., Chen, Y.-T., Tsai, B.-C., Frank Wang, Y.-C., and Sun, M. (2017). No more discrimination: Cross city adaptation of road scene segmenters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1992–2001.

[Chen and Chen, 2017] Chen, Z. and Chen, Z. (2017). Rbnet: A deep neural network for unified road and road boundary detection. In *International Conference on Neural Information Processing*, pages 677–687. Springer.

[Cho et al., 2020] Cho, S. W., Baek, N. R., Koo, J. H., Arsalan, M., and Park, K. R. (2020). Semantic segmentation with low light images by modified CycleGAN-based image enhancement. *IEEE Access*, 8:93561–93585.

[Ciresan et al., 2012] Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems*, pages 2843–2851.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

[Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

[Costea et al., 2017] Costea, D., Marcu, A., Slusanschi, E., and Leordeanu, M. (2017). Creating roadmaps in aerial images with generative adversarial networks and smoothing-based optimization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2100–2109.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE.

[Dash et al., 2021] Dash, A., Ye, J., and Wang, G. (2021). A review of Generative Adversarial Networks (GANs) and its applications in a wide variety of disciplines– From Medical to Remote Sensing. *ArXiv Preprint ArXiv:2110.01442*.

[De Boer et al., 2005] De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

[Deng, 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[Dong et al., 2020] Dong, G., Yan, Y., Shen, C., and Wang, H. (2020). Real-time high-performance semantic image segmentation of urban street scenes. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3258–3274.

[Draegert, 2017] Draegert, J. (2017). Efficient Superpixel Creation in High-resolution Images by Applying a PLANT. *Master Thesis, Freie Universität Berlin*.

[Dubey et al., 2018] Dubey, S. K., Vijay, S., et al. (2018). A review of image segmentation using clustering methods. *Int. J. Appl. Eng. Res*, 13:2484–2489.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Everingham et al., 2015] Everingham, M., Eslami, S., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338.

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[Freemann, 2012] Freemann, B. (2012). Lecture 15: Belief propagation and MRF's. Lecture Notes. Retrieved from http://6.869.csail.mit.edu/fa12/lectures/lecture15mrf/MRF2012.pdf.

[Fritsch et al., 2013a] Fritsch, J., Kuehnl, T., and Geiger, A. (2013a). A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*.

[Fritsch et al., 2013b] Fritsch, J., Kuhnl, T., and Geiger, A. (2013b). A new performance measure and evaluation benchmark for road detection algorithms. In *Intelligent Transportation Systems-(ITSC)*, pages 1693–1700. IEEE.

[Fu and Mui, 1981] Fu, K.-S. and Mui, J. (1981). A survey on image segmentation. *Pattern Recognition*, 13(1):3–16.

[Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

[Gadde et al., 2016] Gadde, R., Jampani, V., Kiefel, M., Kappler, D., and Gehler, P. V. (2016). Superpixel convolutional networks using bilateral inceptions. In *European Conference on Computer Vision*, pages 597–613. Springer.

[Gan et al., 2017] Gan, Z., Chen, L., Wang, W., Pu, Y., Zhang, Y., Liu, H., Li, C., and Carin, L. (2017). Triangle generative adversarial networks. *Advances in Neural Information Processing Systems*, 30.

[Ganin and Lempitsky, 2014] Ganin, Y. and Lempitsky, V. (2014). N^ 4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision*, pages 536–551. Springer.

[Garcia-Garcia et al., 2017] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. *ArXiv Preprint ArXiv:1704.06857*.

[Garcia-Lamont et al., 2018] Garcia-Lamont, F., Cervantes, J., López, A., and Rodriguez, L. (2018). Segmentation of images by color features: A survey. *Neurocomputing*, 292:1–27.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

[Giusti et al., 2013] Giusti, A., Ciresan, D. C., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2013). Fast image scanning with deep max-pooling convolutional neural networks. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 4034–4038. IEEE.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

[Gonzalez, 2009] Gonzalez, R. C. (2009). *Digital image processing*. Pearson Education India.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning (adaptive computation and machine learning series).

[Goodfellow et al., 2020] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

[Hartmann et al., 2017] Hartmann, S., Weinmann, M., Wessel, R., and Klein, R. (2017). Streetgan: Towards road network synthesis with generative adversarial networks.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

[He et al., 2015a] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[He et al., 2015b] He, S., Lau, R. W., Liu, W., Huang, Z., and Yang, Q. (2015b). SuperCNN: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3):330–344.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

[Hochreiter, 1991] Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1).

[Hochreiter, 1997] Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*.

[Ikonomatakis et al., 1997] Ikonomatakis, N., Plataniotis, K., Zervakis, M., and Venetsanopoulos, A. (1997). Region growing and region merging image segmentation. In *Proceedings of 13th International Conference on Digital Signal Processing*, volume 1, pages 299–302. IEEE.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

[Jampani et al., 2016] Jampani, V., Kiefel, M., and Gehler, P. V. (2016). Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4452–4461.

[Ji et al., 2020] Ji, W., Guo, J., and Li, Y. (2020). Multi-head mutual-attention cyclegan for unpaired image-to-image translation. *IET Image Processing*, 14(11):2395–2402.

[Kanopoulos et al., 1988] Kanopoulos, N., Vasanthavada, N., and Baker, R. L. (1988). Design of an image edge detection filter using the Sobel operator. *IEEE Journal of Solid-state Circuits*, 23(2):358–367.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.(dec). *ArXiv Preprint ArXiv:1412.6980*.

[Kohli and Torr, 2007] Kohli, P. and Torr, P. H. (2007). Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088.

[Kohli et al., 2009] Kohli, P., Torr, P. H., et al. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324.

[Krähenbühl and Koltun, 2011] Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Advances in Neural Information Processing Systems*, pages 109–117.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kschischang, 1999] Kschischang, F. (1999). Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):399–431.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

[Ladickỳ et al., 2009] Ladickỳ, L., Russell, C., Kohli, P., and Torr, P. H. (2009). Associative hierarchical CRFs for object class image segmentation. In *2009 IEEE 12th International Conference on Computer Vision*, pages 739–746. IEEE.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., Pereira, F., et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, volume 1, page 3. Williamstown, MA.

[Lateef and Ruichek, 2019] Lateef, F. and Ruichek, Y. (2019). Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2014] Li, H., Zhao, R., and Wang, X. (2014). Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *ArXiv Preprint ArXiv:1412.4526*.

[Li et al., 2009] Li, Q., Wang, B., and Fan, S. (2009). Browse conference publications computer science and engineer. help working with abstracts an improved canny edge detection algorithm. In *2009 Second International Workshop on Computer Science and Engineering proceedings: WCSE*, volume 2009, pages 28–30.

[Lin et al., 2016] Lin, G., Shen, C., van den Hengel, A., and Reid, I. (2016). Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203.

[Liu et al., 2015] Liu, Z., Li, X., Luo, P., Loy, C.-C., and Tang, X. (2015). Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1385.

[Liu et al., 2017] Liu, Z., Li, X., Luo, P., Loy, C. C., and Tang, X. (2017). Deep learning markov random field for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8):1814–1828.

[Lombardi et al., 2005] Lombardi, P., Zannin, M., and Messelodi, S. (2005). Switching models for vision-based on-board road detection. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 67–72. IEEE.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[López et al., 2014] López, M. B., Nieto, A., Boutellier, J., Hannuksela, J., and Silvén, O. (2014). Evaluation of real-time LBP computing in multiple architectures. *Journal of Real-Time Image Processing*, pages 1–22.

[Luc et al., 2016] Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic segmentation using adversarial networks. *ArXiv Preprint ArXiv:1611.08408*.

[Mao et al., 2017] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

[Mirza and Osindero, 2014] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *ArXiv Preprint ArXiv:1411.1784*.

[Mohan, 2014] Mohan, R. (2014). Deep deconvolutional networks for scene parsing. *ArXiv Preprint ArXiv:1411.4101*.

[Moon, 1996] Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60.

[Murphy et al., 1999] Murphy, K. P., Weiss, Y., and Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc.

[Neubert, 2015] Neubert, P. (2015). Superpixels and their application for visual place recognition in changing environments.

[Ning et al., 2005] Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371.

[Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

[Nowozin et al., 2011] Nowozin, S., Lampert, C. H., et al. (2011). Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365.

[O. Russakovsky, 2015] O. Russakovsky, J. Deng, H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. L. F.-F. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Ojala et al., 1994] Ojala, T., Pietikainen, M., and Harwood, D. (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In *Computer Vision and Image Processing*, pages 582–585. IEEE.

[Ojala et al., 1996] Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59.

[Ojala et al., 2002] Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

[Oliveira et al., 2016] Oliveira, G. L., Burgard, W., and Brox, T. (2016). Efficient deep models for monocular road segmentation. In *Intelligent Robots and Systems (IROS)*, pages 4885–4891. IEEE.

[P. Sermanet, 2013] P. Sermanet, D. Eigen, X. Z. M. M. R. F. Y. L. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *ArXiv Preprint ArXiv:1312.6229*.

[Papandreou et al., 2015] Papandreou, G., Chen, L.-C., Murphy, K. P., and Yuille, A. L. (2015). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1742–1750.

[Paszke et al., 2016] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *ArXiv Preprint ArXiv:1606.02147*.

[Pavlidis and Horowitz, 1974] Pavlidis, T. and Horowitz, S. L. (1974). Segmentation of plane curves. *IEEE Transactions on Computers*, 100(8):860–870.

[Prewitt et al., 1970] Prewitt, J. M. et al. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *ArXiv Preprint ArXiv:1511.06434*.

[Rafael, 1992] Rafael, C. (1992). Gonzalez, and richard e. woods. *Digital Image Processing*, page 793.

[Ren and Malik, 2003] Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *ICCV*, volume 1, pages 10–17.

[Ribani and Marengoni, 2019] Ribani, R. and Marengoni, M. (2019). A survey of transfer learning for convolutional neural networks. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pages 47–57. IEEE.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

[Romera et al., 2017] Romera, E., Alvarez, J. M., Bergasa, L. M., and Arroyo, R. (2017). Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image Computing and Computer-Assisted Intervention*, pages 234–241. Springer.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

[Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, (23):3.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. *Biometrika*, 71(599-607):6.

[Saeed, 2020] Saeed, J. N. (2020). A survey of ultrasonography breast cancer image segmentation techniques. *Academic Journal of Nawroz University*, 9(1):1–14.

[Sahoo et al., 1988] Sahoo, P. K., Soltani, S., and Wong, A. K. (1988). A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260.

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[Schwing and Urtasun, 2015] Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *ArXiv Preprint ArXiv:1503.02351*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*.

[Song and Kim, 2020] Song, A. and Kim, Y. (2020). Semantic segmentation of remote-sensing imagery using heterogeneous big data: International society for photogrammetry and remote sensing potsdam and cityscape datasets. *ISPRS International Journal of Geo-Information*, 9(10):601.

[Springenberg et al., 2014] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *ArXiv Preprint ArXiv:1412.6806*.

[Sturgess et al., 2009] Sturgess, P., Alahari, K., Ladicky, L., and Torr, P. H. (2009). Combining appearance and structure from motion features for road scene understanding. In *BMVC-British Machine Vision Conference*. BMVA.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[Szeliski et al., 2006] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for markov random fields. In *European Conference on Computer Vision*, pages 16–29. Springer.

[Tanaka, 1999] Tanaka, T. (1999). A theory of mean field approximation. In *Advances in Neural Information Processing Systems*, pages 351–360.

[Ulku and Akagündüz, 2022] Ulku, I. and Akagündüz, E. (2022). A survey on deep learning-based architectures for semantic segmentation on 2d images. *Applied Artificial Intelligence*, pages 1–45.

[Verma et al., 2010] Verma, A., Banerji, S., and Liu, C. (2010). A new color SIFT descriptor and methods for image category classification. In *International Congress on Computer Applications and Computational Science*, pages 4–6.

[Vezhnevets et al., 2012] Vezhnevets, A., Ferrari, V., and Buhmann, J. M. (2012). Weakly supervised structured output learning for semantic segmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–852. IEEE.

[Vincent and Soille, 1991] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(06):583–598.

[Wang and Deng, 2018] Wang, M. and Deng, W. (2018). Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153.

[Weiss and Freeman, 2001] Weiss, Y. and Freeman, W. T. (2001). On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744.

[Weller and Westneat, 2019] Weller, H. I. and Westneat, M. W. (2019). Quantitative color profiling of digital images with earth mover's distance using the R package colordistance. *PeerJ*, 7:e6398.

[Widrow, 1962] Widrow, B. (1962). Generalization and information storage in networks of adaline neurons. *Self-organizing systems*, pages 435–461.

[Williams and Rasmussen, 1996] Williams, C. K. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, pages 514–520.

[Withey and Koles, 2008] Withey, D. J. and Koles, Z. J. (2008). A review of medical image segmentation: methods and available software. *International Journal of Bioelectromagnetism*, 10(3):125–148.

[Wu et al., 2008] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.

[Yeboah et al., 2018] Yeboah, Y., Yanguang, C., Wu, W., and Farisi, Z. (2018). Semantic scene segmentation for indoor robot navigation via deep learning. In *Proceedings of the 3rd International Conference on Robotics, Control and Automation*, pages 112–118.

[Yedidia et al., 2003] Yedidia, J. S., Freeman, W. T., Weiss, Y., et al. (2003). Understanding belief propagation and its generalizations. *Exploring Artificial Intelligence in the New Millennium*, 8(236-239):0018–9448.

[Yu, 2005] Yu, S. X. (2005). Segmentation induced by scale invariance. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 444–451. IEEE.

[Zaitoun and Aqel, 2015] Zaitoun, N. M. and Aqel, M. J. (2015). Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806.

[Zhang et al., 2019] Zhang, Y., Li, X., and Zhang, Q. (2019). Road topology refinement via a multi-conditional generative adversarial network. *Sensors*, 19(5):1162.

[Zhao et al., 2017] Zhao, B., Feng, J., Wu, X., and Yan, S. (2017). A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 14(2):119–135.

[Zhao et al., 2018] Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J. (2018). ICNET for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420.

[Zheng et al., 2015] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.

[Zhou et al., 2016] Zhou, H., Zhang, J., Lei, J., Li, S., and Tu, D. (2016). Image semantic segmentation based on FCN-CRF model. In *2016 International Conference on Image, Vision and Computing (ICIVC)*, pages 9–14. IEEE.

[Zhu et al., 2016] Zhu, H., Meng, F., Cai, J., and Lu, S. (2016). Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232.

[Zohourian et al., 2018a] Zohourian, F., Antic, B., Siegemund, J., and Meuter, M. (2018a). Method for the semantic segmentation of an image. US Patent App. 15/949,246.

[Zohourian et al., 2018b] Zohourian, F., Antic, B., Siegemund, J., Meuter, M., and Pauli, J. (2018b). Superpixel-based Road Segmentation for Real-time Systems using CNN. In *VISIGRAPP (5: VISAPP)*, pages 257–265.

[Zohourian and Pauli, 2022a] Zohourian, F. and Pauli, J. (2022a). Coarse-to-Fine Semantic Road Segmentation Using Super-Pixel Data Model and Semi-Supervised Modified CycleGAN. *Journal of Image and Graphics*, 10(4):132–144.

[Zohourian and Pauli, 2022b] Zohourian, F. and Pauli, J. (2022b). Enhancement of the super pixel-CNN based road segmentation using cycle consistent adversarial network. In *Fourteenth International Conference on Machine Vision (ICMV 2021)*, volume 12084, page 120840D.

[Zohourian et al., 2018c] Zohourian, F., Siegemund, J., Meuter, M., and Pauli, J. (2018c). Efficient fine-grained road segmentation using superpixel-based CNN and CRF models. In *International Conference on Pattern Recognition and Artificial Intelligence ICPRAI 2018*, pages 512–517. CENPARMI, Centre for Pattern Recognition and Machine Intelligence Concordia University, Montréal, Canada.

[Zohourian et al., 2022] Zohourian, F., Siegemund, J., Meuter, M., and Pauli, J. (2022). Efficient fine-grained road segmentation using superpixel-based CNN and CRF models. *ArXiv Preprint ArXiv:2207.02844*.

Appendix I: Additional Segmentation Examples

In this appendix, we provide additional segmentation examples related to the techniques discussed in Chapter 3. These examples serve to further illustrate the results and outcomes of our methods. Specifically, we showcase additional samples of road segmentation achieved through our *SVM* methods, as explained in Section 3.4.2, and also present more samples of road segmentation utilizing our SP-CNN method described in Section 3.4.2. Below are the additional segmentation examples:

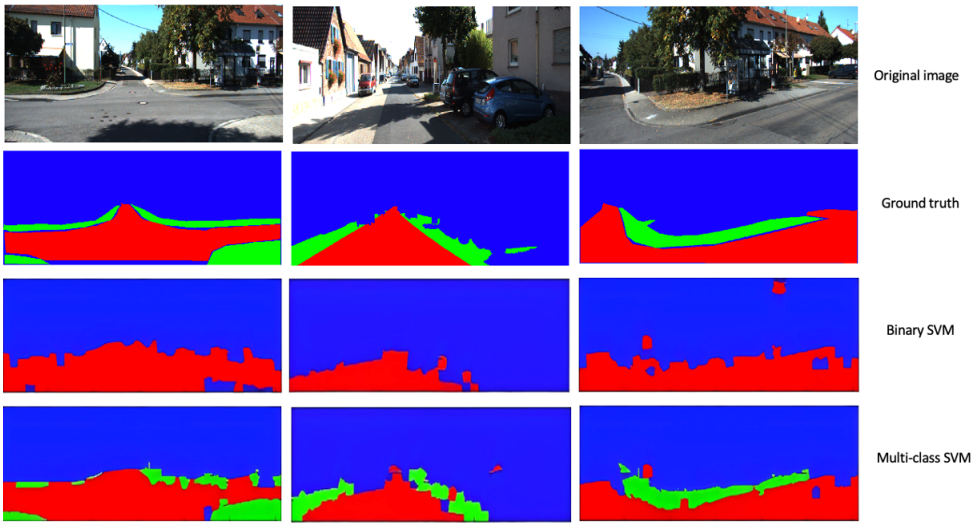


Figure 1: Additional samples of road segmentation using our *SVM* methods from Section 3.4.2.

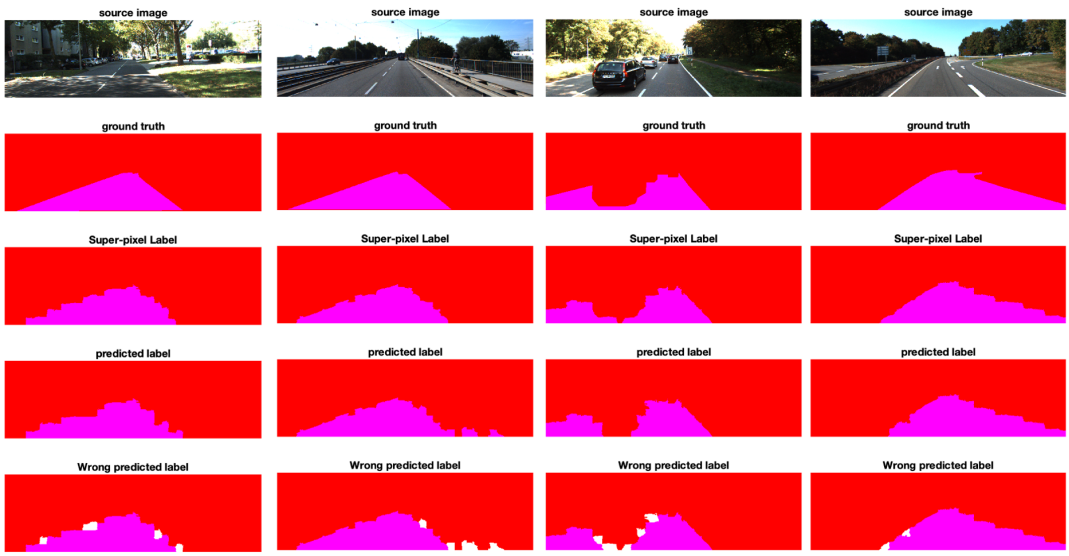


Figure 2: Additional samples of road segmentation using our SP-CNN method from Section 3.4.2.

Appendix II: Detailed Generator and Discriminator Architectural Diagrams

In this second appendix, we delve into the intricate architectural details of the components discussed in Chapter 5. These detailed diagrams provide a comprehensive view of the inner workings of our proposed Generators $G_{X \to Y}$, and $F_{Y \to X}$ and corresponding discriminators D_Y, D_X, each split across multiple pages for clarity. The diagrams presented here serve as supplementary material, offering readers a closer look at the design and structure of these crucial elements within our research.

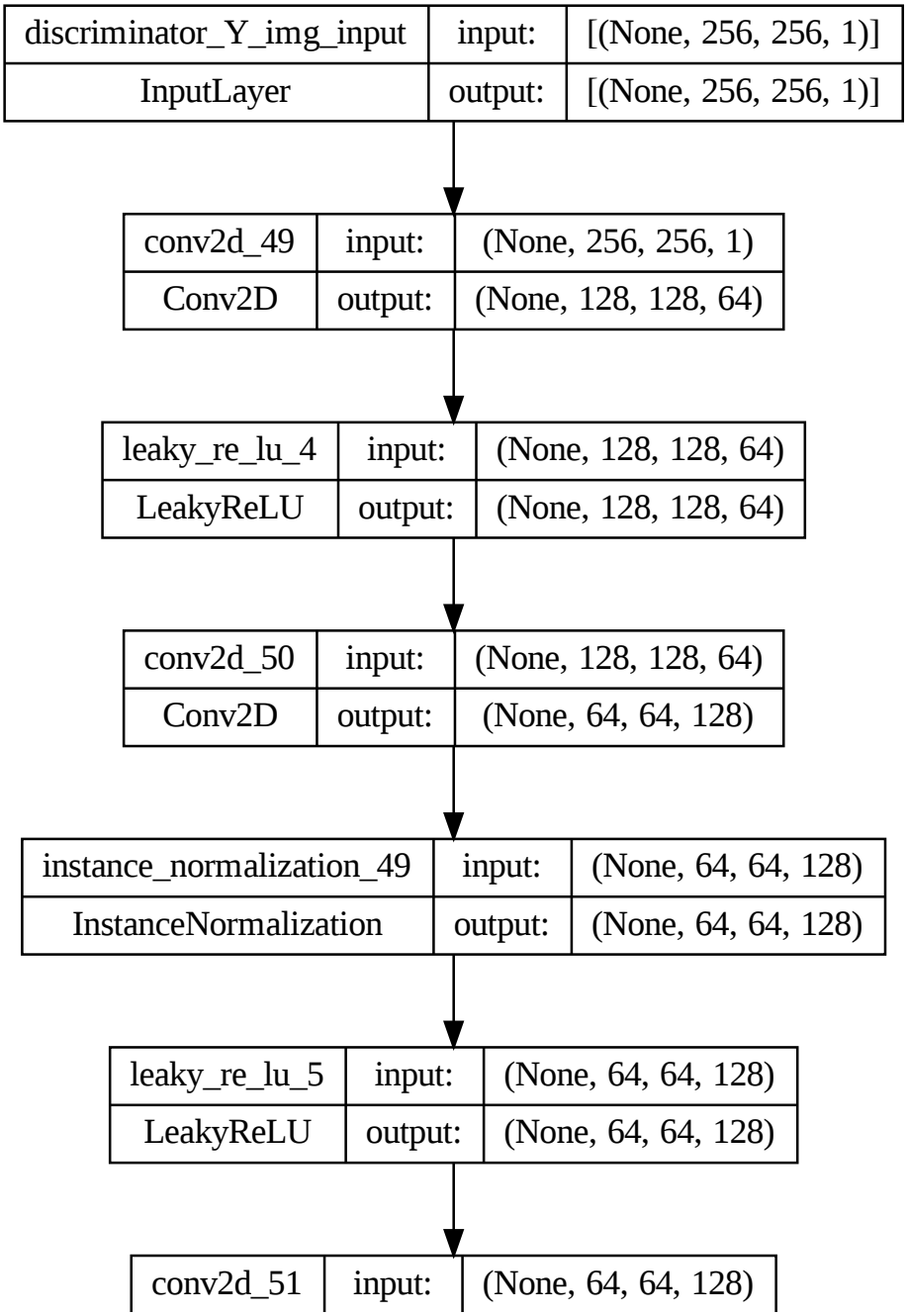


Figure 1: Architecture of our proposed Discriminator D_Y (part1).

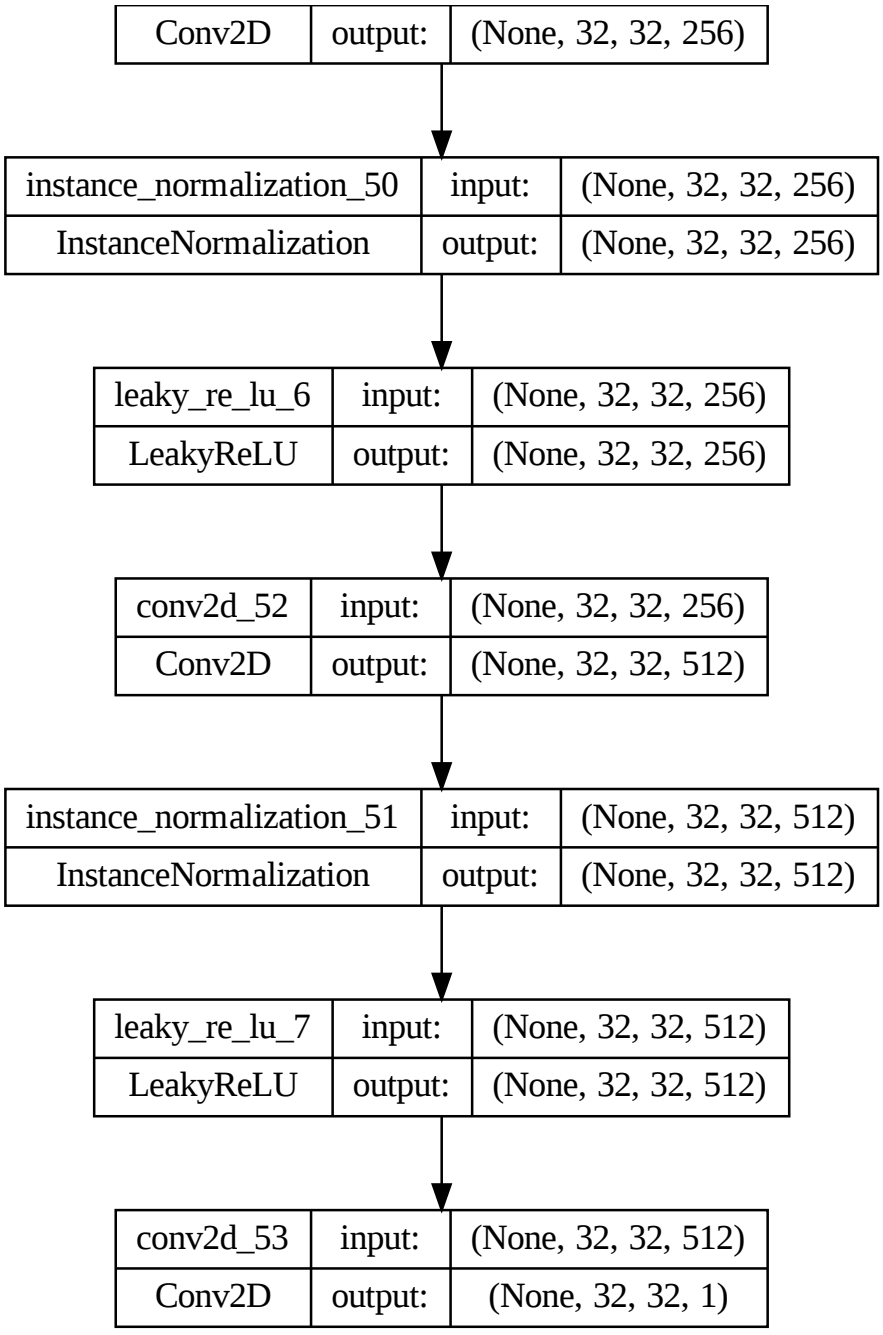


Figure 2: Architecture of our proposed Discriminator D_Y (part2).

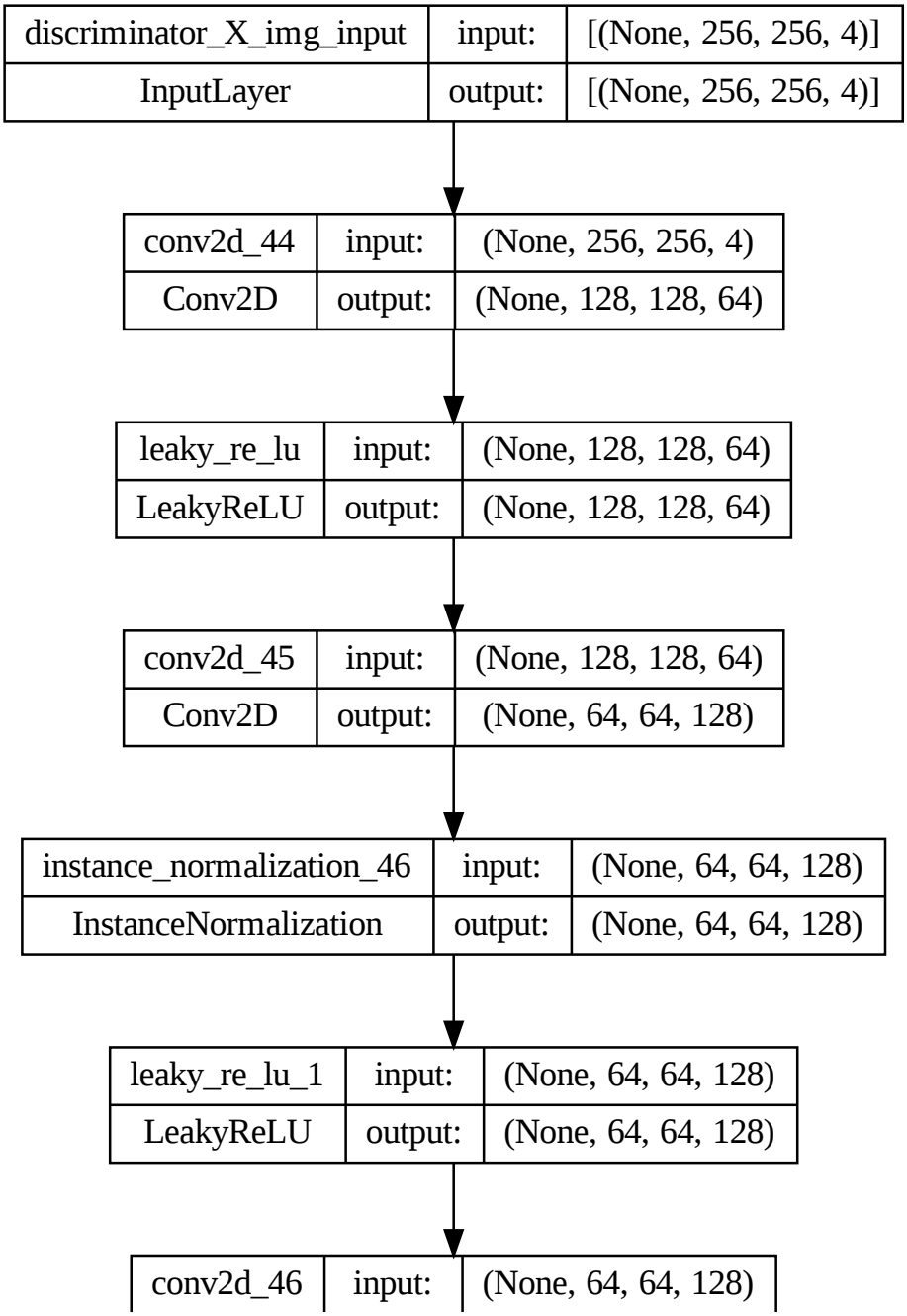


Figure 3: Architecture of our proposed Discriminator D_X (part1).

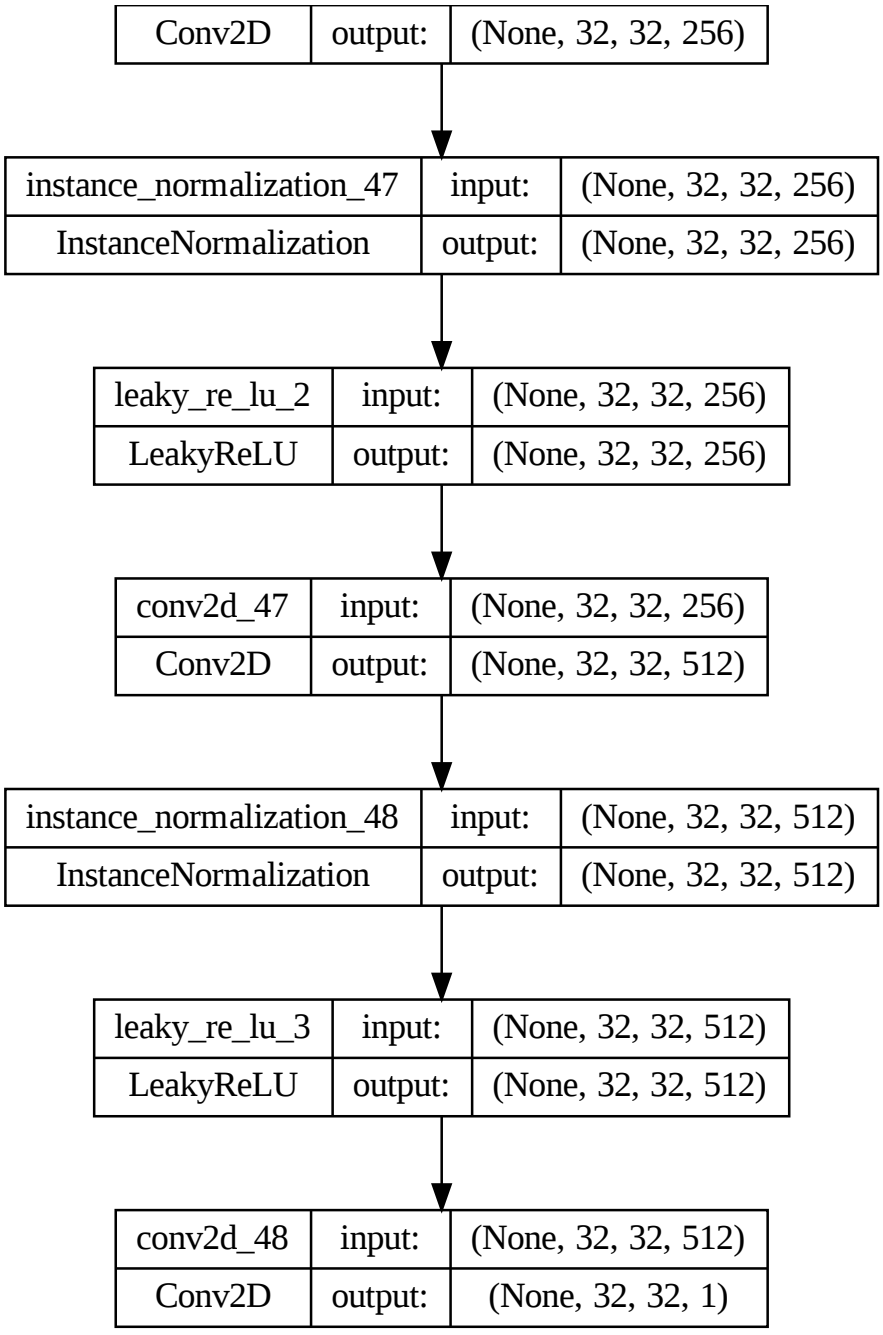


Figure 4: Architecture of our proposed Discriminator D_X (part2).

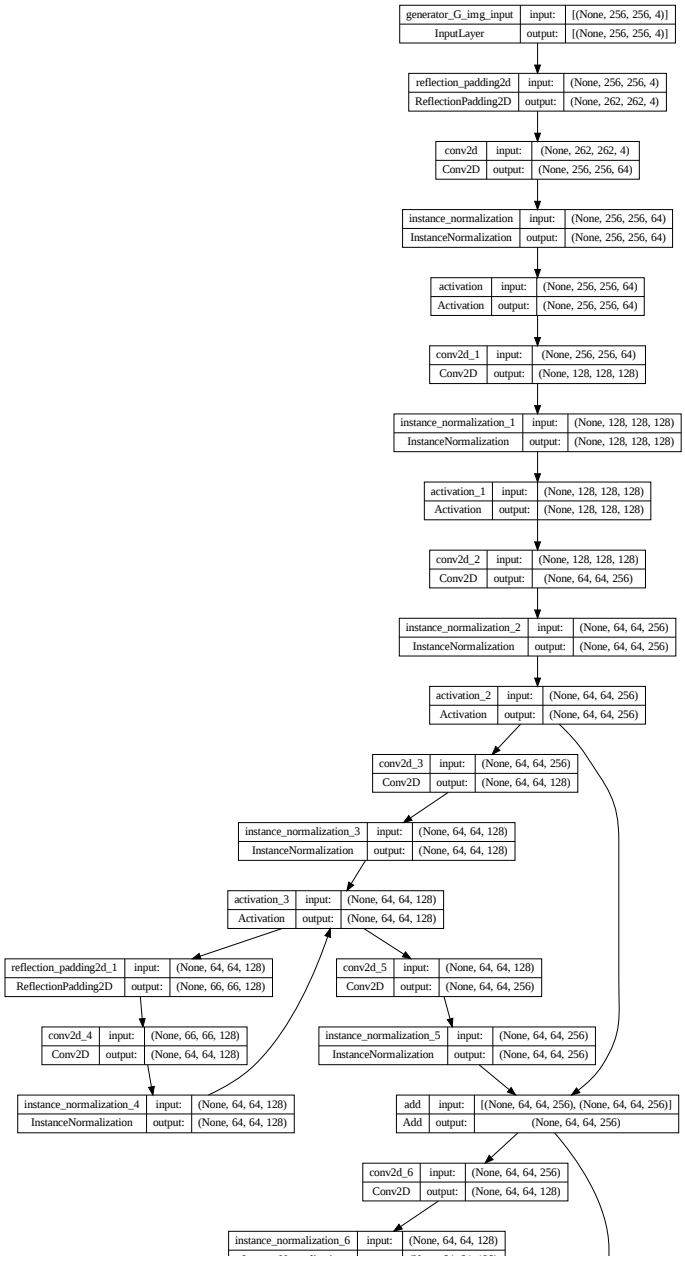


Figure 5: Architecture of our proposed Generator $G_{X \to Y}$ (part1).

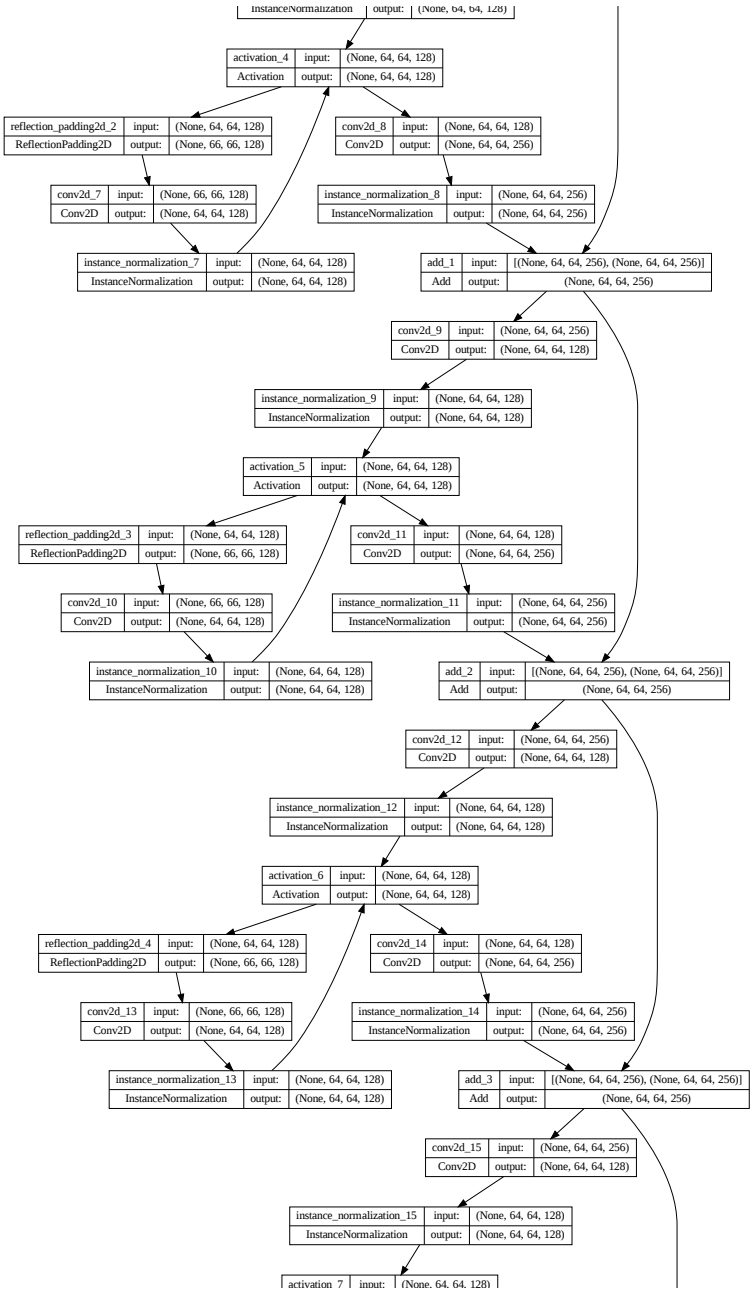


Figure 6: Architecture of our proposed Generator $G_{X \to Y}$ (part2).

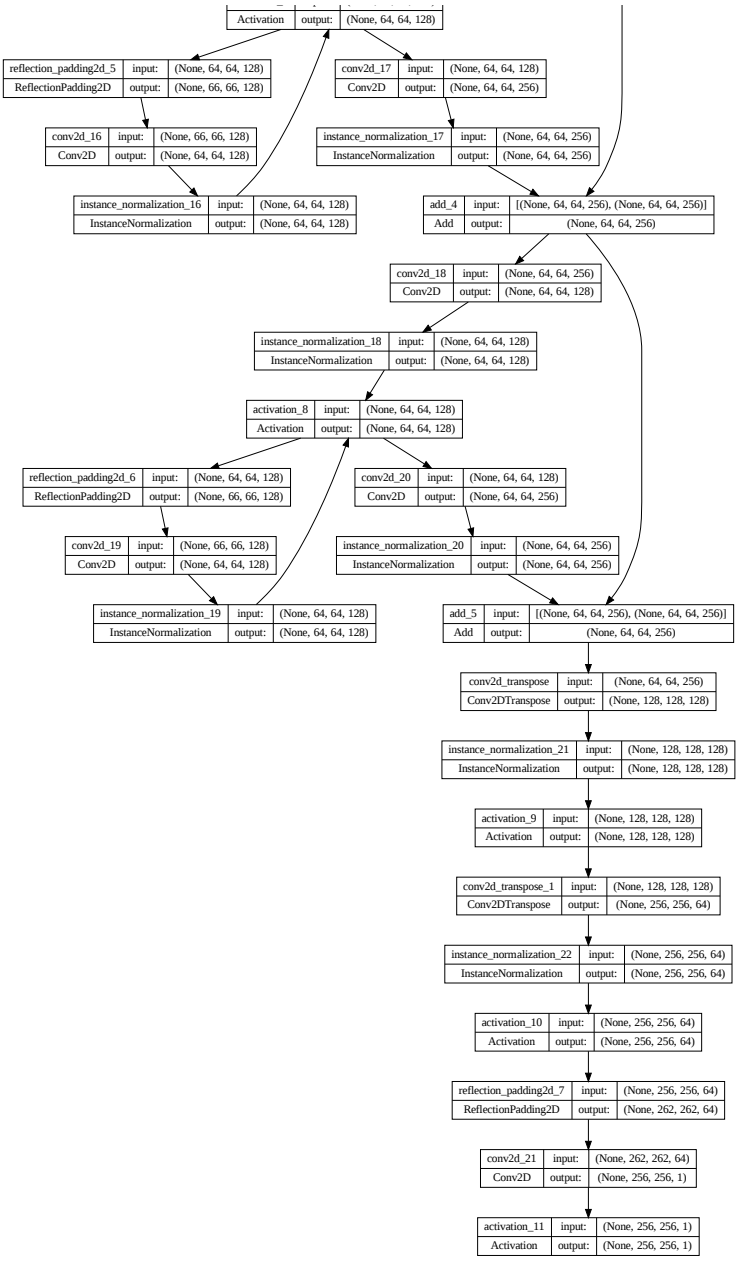


Figure 7: Architecture of our proposed Generator $G_{X \to Y}$ (part3).

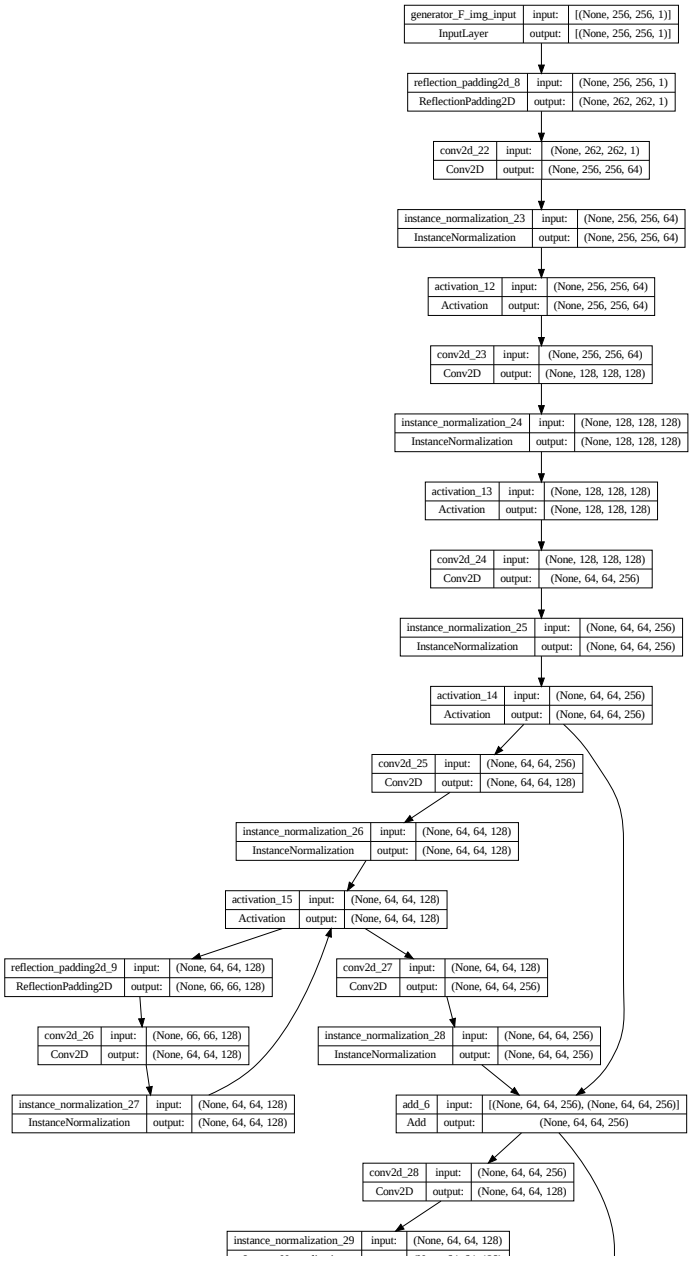


Figure 8: Architecture of our proposed Generator $F_{Y \to X}$ (part1).

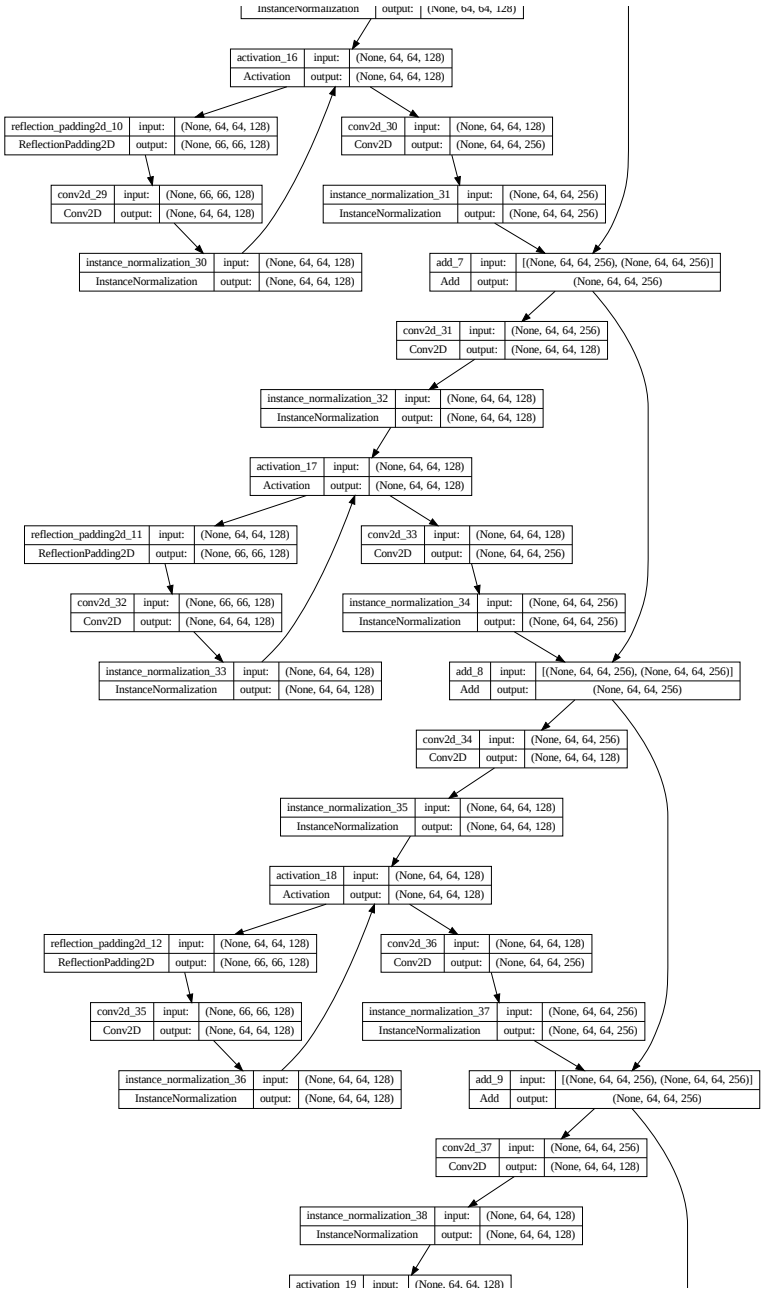


Figure 9: Architecture of our proposed Generator $F_{Y \to X}$ (part2).

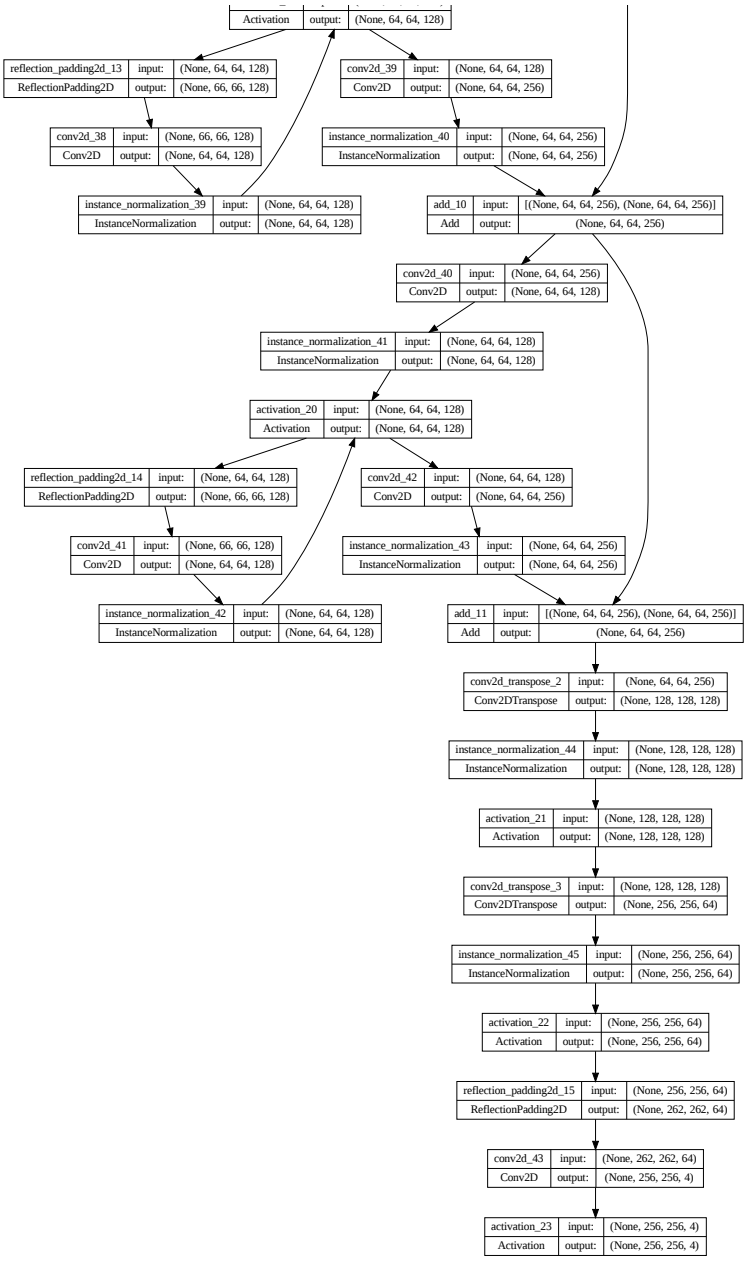


Figure 10: Architecture of our proposed Generator $F_{Y \to X}$ (part3).

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Ich versichere weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingesetzt wurde.

Düsseldorf, 17/04/2024

Ort, Datum *Unterschrift*

Versicherung an Eides Statt

Ich versichere an Eides Statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen übernommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe.
Ich versichere an Eides Statt, dass ich die vorgenannten Angaben nach bestem Wissen und Gewissen gemacht habe und dass die Angaben der Wahrheit entsprechen und ich nichts verschwiegen habe. Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 163 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Düsseldorf, 17/04/2024

Ort, Datum

Unterschrift