Deep Learning Methods for Video Smoke Detection

Dissertation

University of Duisburg-Essen

Andreas Wellhausen

2021

Deep Learning Methods for Video Smoke Detection

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Elektrotechnik und Informationstechnik
der Universität Duisburg-Essen
genehmigte Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften

von
Andreas Wellhausen
aus
Hameln

Datum der Disputation:
20.10.2021

Gutachter:
Prof. Dr. Ingolf Willms
Prof. Dr. Thomas Kaiser

Abstract

Video Smoke Detection is a promising solution to detected fires in buildings with high ceilings (e.g., factories, warehouses, train stations, and tunnels) or outdoor areas (e.g., landing stripes, harbors, and pedestrian areas). It can detect smoke very fast and prevent higher human or property damage.

These benefits are the reason why an increasing amount of research groups and companies are aiming to develop reliable algorithms for Video Smoke Detection.

In classic approaches, physical or visual characteristics of smoke are identified and extracted by ordinary Computer Vision algorithms to distinguish smoke from non smoke events. These approaches require substantial limitations to the field of application to assure that smoke behaves as expected.

Furthermore, Video Smoke Detection suffers from high false alarm rates, such that no fully automatic smoke detection is possible and an alarm candidate has to be checked by humans.

Due to the success of artificial intelligence in object detection, research in Video Smoke Detection shifts more and more to apply Deep Learning methods.

In this thesis, it is shown that Deep Learning methods outperform classical Computer Vision algorithms by far and can enable fully automatic Video Smoke Detection systems. Several state of the art Deep Learning methods are investigated successfully concerning performance and computing complexity. This analysis includes single frame approaches based on convolutional neural networks and temporal approaches utilizing 3D convolutions or recurrent networks. It turns out that temporal information is crucial for Deep Learning methods in Video Smoke Detection. Temporal input, like difference or optical flow of two consecutive images also improves the results.

Among all investigated methods, the *i3D*, a network using 3D convolutions, in combination with difference images performs best. It detects smoke very fast, in many situations even faster than human. Furthermore, the computing complexity is reduced by a custom approach to 1% while maintaining 92% of the *i3D*'s performance. This is valuable, when meeting hardware restrictions on the target platform.

Acknowledgements

The proposed thesis is the result of my three years research work at Bosch Building Technologies GmbH in coorperation with Department of Nachrichtentechnische Systeme at the University of Duisburg/Essen.

Ein besonderer Dank geht an Dr. Anton Stadler und Dr. Tjark Windisch von Bosch Building Technologies GmbH, die mich jederzeit mit Rat und Tat unterstützt haben. Sie haben mich ermutigt und es mir ermöglicht, meiner Kreativität freien Lauf zu lassen. Außerdem haben Sie mir durch ihr Feedback geholfen in der Spur zu bleiben.
Ich möchte auch Prof. Dr. Ingolf Willms und Dr. Thorsten Schultze von der Universität Duisburg/Essen dafür danken, dass sie die akademische Betreuung dieser Arbeit übernommen haben.
Desweiteren auch vielen Dank an meine Bosch Kollegen aus Hildesheim und Grasbrunn für die vielen anregenden Diskussionen, die es mir ermöglicht haben auch mal über den Tellerrand der vorliegenden Arbeit hinauszuschauen.

Danke auch an meine Eltern, die mich während meines gesamten Ausbildungsweges begleitet haben.
Zuletzt möchte ich noch meiner Frau Birthe danken, die in der finalen Phase meiner Dissertation sehr zurückstecken musste, mich aber so unterstützt hat, dass ich mich 100% auf meine Arbeit konzentrieren konnte.

Abbreviations

3D-CNN Spatial and Temporal Convolutional Neural Network.

AM Ambient smoke.

BB Bounding Box.

BGF Background Flickering smoke.

BP Back Propagation.

BPTT Back Propagation Trough Time.

CNN Convolutional Neural Network.

ConvLSTM Convolutional Long Short Term Memory Network.

CV Computer Vision.

DE Directed Expanding smoke.

Diff Difference of two consecutive images..

DL Deep Learning.

FC Fully Connected Layer.

Flow Optical flow between two consecutive images.

GRU Gated Rectified Unit.

i3D inflated 3D.

IID identical and independent distributed.

LSTM Long Short Term Memory Network.

ML Machine Learning.

NN Neural Network.

ReLU Rectified Linear Unit.

RGB R=red, G=green, B=blue 24bit image representation.

RNN Recurrent Neural Network.

ROC Receiver Operating Curve.

RPN Region Proposal Network.

SGD Stochastic Gradient Descent.

SotA State of the Art.

SSD Single Shot Detection.

UE Undirected Expanding smoke.

VSD Video Smoke Detection.

YOLO You Only Look Once.

List of Symbols

X	Set of observed samples.
Y	Set of labels.
x	One observed sample.
y	One label.
f	Arbitrary function.
$\mathcal{N}$	Arbitrary neural network.
β	Parameters of a neural network.
l	Number of kernels in a layer.
K	Convolutional kernel.
k	Spatial size of a convolutional kernel or a pooling layer.
c	Number of feature channels of a convolutional kernel or a tensor.
s	Stride of a convolutional kernel.
κ	Arbitrary activation function.
I	Input tensor of a layer.
h	Height of a tensor.
w	Width of a tensor.
O	Output tensor of a layer.
$*$	Discrete convolution.
σ	Sigmoid function.
f_t	Forget gate of an LSTM.
i_t	Input gate of an LSTM.
o_t	Output gate of an LSTM.
c_t	Carry state of an LSTM.
r	Receptive field of a covolutional neural network.
$\mathcal{L}$	Loss function for the optimization of a neural network.

ρ	Learning rate for SGD.
μ	Momentum of SGD.
b	Number of Batches.
TPR	True Positive Rate.
FPR	False Positive Rate.
λ	Sensitivity parameter of VSD system to generate ROC curve.
AUC	Area under the ROC curve.
θ	Maximal FPR for restricted ROC curve.
T	Length of a sequence.
D_{Speed}	Detection speed to measure the performance of a VSD system.
r_v	Vertical size of the input image.
r_h	Horizontal size of the input image.
v	Smoke velocity.
η	Opening angle of the camera.
S	Sequences consisting of consecutive frames.
B	Set of bounding boxes.
C	Cells of the grid, which is blended on the image.
M	Binary mask for existence of smoke in each cell.
$\mathcal{A}$	DL System, which detects smoke in a video sequence.
$\mathcal{F}$	Feature extractor.
$\mathcal{C}$	Classifier.
P	Probability map of smoke in each cell.
$\mathcal{B}$	Subset of training set, which forms a batch.
δ	Decay rate, with which the learning rate is reduced each epoch.
τ	Dropout rate, i.e. the fraction of the randomly chosen features, which are set to zero.
ω	Weight of regularization part of the loss.
γ	Time constant of temporal accumulation.
MAC	Multiplications and accumulations. A measure for the computational complexity of a neural network.
α	Parameter to define the kernel depths of MobileNetV2.

Contents

Chapter 1

Introduction

Firstly, a motivation and purpose of Video Smoke Detection (VSD) is given in Section 1.1. In Section 1.2 the requirements for VSD algorithms are defined. Section 1.3 discusses the state of the art (SotA) in VSD. The contribution of this thesis beyond the SotA is described in Section 1.4. Finally, an overview of the thesis' structure is presented in Section 1.5.

1.1 Video Smoke Detection (VSD)

1.2 billion dollars losses, 16 deaths and 273 injuries annually, all due to fires in industrial and manufacturing buildings - these are the annual figures in the United States [1]. Smoke detection systems are installed to reduce these death, injuries, and costs. A classical smoke detector is placed under the ceiling and triggers an alarm due to the optical, thermal, or chemical properties of smoke. This process is not optimal for large factories and buildings with high ceilings for two main reasons. Firstly, smoke needs time to reach the detector, which makes smoke detection very slow. On the other hand, a single detector monitors only a tiny fraction of a room. Both problems are solved by VSD. VSD is the solution in fields of application, where utilization of ordinary smoke detectors is restricted, or a fast smoke detection is crucial to prevent personal injury and material damage. VSD can also be applied in outdoor scenarios, where currently no approach is available.

A camera can observe a larger room and detects smoke just as it starts to spread. Additionally, video material can be transmitted to the fire brigade before it arrives,

giving them an advantage in analyzing the situation. As smoke detection decides on life and death, reliable VSD Algorithms are of humane and economic importance.
There are three ways of detecting smoke in a video surveillance area. One is to let a person observe the recordings in real-time, which is expensive and a demanding job since it requires great attention over several hours. The second way is to develop a Computer Vision (CV) algorithm, which enables a fully automatic smoke detection. To the current state of the art (SotA), this method suffers from a not acceptable amount of false alarms. Therefore the third way is usually applied: The CV based VSD algorithm detects possible smoke candidates, transmits them to a central control unit, which decides, whether it is a real smoke alarm or not.
Nevertheless, the overall goal is to have a fully automatic VSD using CV methods. Deep Learning (DL) methods applied on VSD are promising to make one step ahead of this goal since they reach remarkable results in other object detection problems and outperform all other classic CV methods.

1.2 Requirements for VSD

In this thesis, cameras are installed to observe an arbitrary area, in which smoke should be detected. Infrared cameras or other sensors delivering 2D images are not considered. The recordings of each camera are processed independently and not merged.
It is assumed that VSD is applied in a typical surveillance scenario, i.e., the camera is fixed and does not move except some slightly shaking caused by wind or vibrations in the environment. Relevant are all smoke recordings, in which humans can detect smoke for sure by watching the sequence. Smoke is not recognizable by a human when the event is too far away from the camera, completely covered by an object or illuminated weakly. There are no further restrictions on the environment, weather, or illumination conditions. Events, which can only be distinguished from dangerous smoke by understanding the context, e.g., steam leaving funnels or dust, are not considered.
A fully automatic VSD system is assessed to its ability to detect smoke as fast as possible while being robust against false alarms, but a late smoke detection is better

than no smoke detection. The maximum valid time for smoke detection is set to 90 seconds after the first smoke is human visible. A maximum distance to the smoke event is not defined. The most events considered in this thesis are 5 to 30 meters apart from the camera. The exact location of smoke in the frame is not relevant.

1.3 State of the Art

In classic approaches, the development of VSD algorithms is done in two steps: Firstly, temporal and spatial features based on the visual characteristics of smoke are identified. Such features transform the problem of VSD in an easier to handle lower-dimensional domain. In the best case, smoke is separable from disturbance events in this domain. Typical smoke characteristics for feature development are color [2,3], static texture [4–11], dynamic textures [12–16], moving and growing behavior [17–25].

In recent years Deep Learning methods are also investigated for VSD. The research starts by observing single frame methods. It is shown that one is able to detect smoke by using convolutional neural networks (CNNs) [26]. Different SotA CNN architectures are compared like *VGG*, *ResNet*, *Inception* and *Xception*. The experimental result is that *Xception* works best [27].

Temporal approaches are also investigated. A two-stream approach using two networks one for RGB and one for optical flow is analyzed by [28]. The resulting features are merged to classify smoke. A public dataset of smoke and non smoke sequences crops is constructed and a combination of CNN and recurrent neural network (RNN) are observed by [29] with promising results. This dataset is also used for SotA comparison in this thesis. But this dataset is not sufficient to evaluate the algorithms in the scope of a full VSD system.

The lack of data is a big issue when utilizing DL methods for VSD. To overcome this issue [30] enlarges their dataset by artificial smoke data and show that this could increase the performance. They are investigating single frames only.

1.4 Contribution of this Thesis

In this thesis it is shown, that DL methods outperform classical CV methods by far and enables fully automatic VSD systems. In contrast to the current SotA the DL methods are embedded and evaluated as a full VSD system, i.e. it is assumed that a camera delivers real time surveillance recordings and the algorithm responds with a smoke prediction using past information only. Several new concepts concerning data preparation and evaluation are developed and DL techniques are compared.
Data are crucial to develop and compare DL algorithms. This thesis is supported by the *Bosch Building Technology GmbH*, which provides an exhaustive dataset of smoke and non smoke (negative) recordings in surveillance scenarios, about 350 hours of video material. These data are structured and labeled for DL purposes.
This thesis compares several VSD algorithm. The algorithms should be evaluated due to their ability to be a standalone VSD system. This means that the algorithms are evaluated on real surveillance sequences containing smoke and negative periods. For evaluation and comparison of the VSD algorithms a measure is developed, which monitors the requirements of Section 1.2. It is called *Detection Speed* and measures the ability of an algorithm to detect smoke as fast as possible while being robust against false alarms.
Using this data and evaluation concept, several VSD approaches, which are already investigated for VSD in literature are revisited: An algorithm based on traditional CV methods using an accumulation of optical flow over time. A single frame based CNN using RGB.
These algorithms act as baselines to compare them to algorithms utilizing new DL ideas. The first is to use transfer learning for the CNN with RGB input. The second is to add temporal information, which is done in two ways: Firstly using temporal input, i.e., optical flow and image differences. Secondly, to investigate DL architectures with the ability to learn temporal information. One is a combination of CNNs and RNNs, which first extracts spatial information, of which temporal features are derived. The other one is a SotA architecture using 3D convolutions (3D-CNN). Furthermore, a custom approach is presented, which uses the smoke probabilities of a CNN as an input and accumulates them over time. Also, the approaches using

temporal DL architectures are investigated with input of image differences. It turns out that the 3D-CNN with differences of images as input works best.
It is shown that all the approaches newly introduced in this thesis outperform the classic VSD algorithms and Deep Learning VSD approaches by far.
DL architectures are very resource consuming. So especially when hardware restrictions occur, the application of these algorithms in real-time is limited. Therefore architectures, which reduce the computing complexity, are investigated. Firstly SotA concepts for complexity reduction of CNNs and 3D-CNNs are analyzed. Secondly, a custom architecture based on convolutional RNNs and a custom concept for further reducing the complexity of 3D-CNNs is observed, and it is shown that the complexity can significantly be reduced with less performance decrease.

1.5 Structure of this Thesis

This thesis starts with an overview of the basics for algorithm development using DL methods and delimitates it from classical approaches in Chapter 2.
Chapter 3 describes a structuring concept for data to develop and compare DL based VSD algorithm. Additionally, a measure to compare the VSD is defined, and a VSD algorithm using classic CV methods is developed and evaluated.
A template for all DL architectures and the method, how they are trained is described in Chapter 4.
This template is applied to a CNN using RGB images as input in Chapter 5. Furthermore, temporal DL methods are developed, compared, and analyzed in the scope of VSD. Additionally, these architectures are investigated on a public dataset.
In Chapter 6 a measure to compare the model complexity is defined, and several methods to reduce the model complexity are observed.
Finally, a summary, results, and open questions are given in Chapter 7.

Chapter 2

Basics

This chapter deals with algorithm development for VSD and introduces, how this could be done using DL methods. First, an overview of how VSD algorithms are developed in literature is given in Section 2.1. Detailed information about classic CV algorithm and which techniques are used are not relevant for the proposed thesis. A comprehensive overview is given in [31]. Rather interesting for this thesis is the general concept, how features are designed and passed a decision function to predict smoke. This classic concept is delimited to DL based methods.
In Section 2.2 the theoretical DL background is described. The thesis relevant DL modules CNNs, 3D-CNNs, and RNNs are explained. Furthermore, it is discussed, what challenges occur, when training and evaluating such DL modules.
An overview of the SotA in object detection and sequence classification is presented in Section 2.3.

2.1 VSD Algorithm Development

In classical approaches features for VSD are designed by hand with much knowledge about visual and physical smoke behavior, this approach is called handcrafted feature extraction. A VSD algorithm based on such features is called expert system.
Secondly, rules for an alarm decision in the feature domain are derived. A decision function models these rules, and the goal is to get a good approximation of this decision function. An architecture and a related set of parameters defines such a decision function. Both are unknown and have to be determined. One realization

of an architecture with parameters is called model. Two ways are applied to approximate such a decision function by a model. One is to derive the rules based on physical smoke properties. The other way is to observe sample events, investigate the feature domain of those, and derive a model statistically using machine learning (ML) methods like logistic regression, support vector machine or bayesian classifier. Figure 2.1 gives an overview.

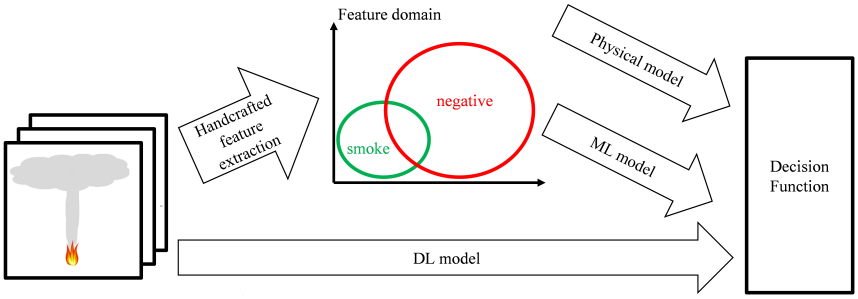
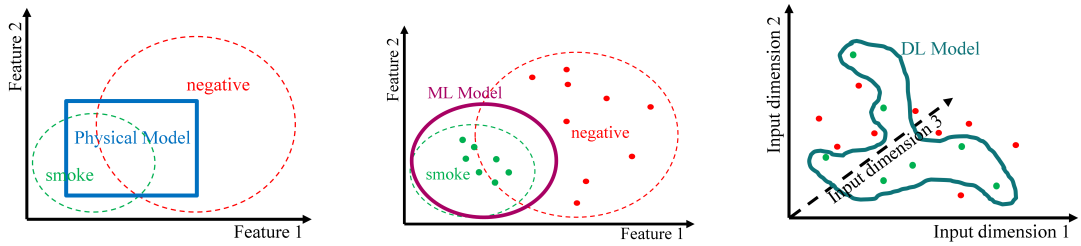


Figure 2.1: Development workflow of VSD algorithms. The raw images are transformed into the feature domain, in which a decision function separates smoke from negatives. Using DL method the decision function is directly derived from the raw data.

Two problems could occur using handcrafted features and a physical model or ML based decision function. The first is that smoke and negative events are not perfectly separated in the feature domain, which is indicated by the overlap of the green and red circle in Figure 2.1. Secondly, that the decision function does not fit the smoke region adequately in feature domain (Figures 2.2a and 2.2b). In classic algorithm development, these problems are tackled separately in iterations to improve the VSD system, which leads to very complex and hard to maintain algorithms.
In contrast, DL approaches do not necessarily require knowledge of experts. The decision function is directly derived from the raw data, without or with low-level handcrafted feature extraction. Strictly speaking, DL is a specific ML technique containing neural networks since it also uses real-world data and statistical methods to separate smoke from negative events. The difference is that the effort for feature design is much reduced or completely skipped, and the time for sample generation, architecture and parameter selection is increased. By reducing the handcrafted feature design, the feature domain keeps very high or unreduced compared to the original problem (Figure 2.2c). Therefore it is said that DL methods automatically



(a) (b) (c)

Figure 2.2: Different types of decision functions. (a) shows a physical model in the feature domain. All input within the blue box is classified as smoke. (b) illustrates ML in feature domain. The green and red dots are the sample data to determine the model. All input in the violet box is classified as smoke. (c) shows a DL model on raw data. All samples in the turquoise region are classified as smoke.

derive features and decide based on these features.

The weakness of DL models is that they require much data, which should be representative of all real-world smoke and negative events. ML models require data for training depending on the complexity of the feature domain, usually a very huge amount to get good results and additionally for testing. In contrast, physical models need data for testing only.

2.2 Deep Learning Basics

The key tool for single frame DL in CV are convolutional neural networks (CNN), which are used to learn spatial features for defined tasks automatically. These CNNs are explained at first.

Secondly, two ways, how to deal with image sequences are described: 3D-CNNs and recurrent neural networks RNN. 3D-CNNs are extended versions of CNNs to the temporal dimension. RNNs carry temporal information from frame to frame. Furthermore, the receptive field and how to determine it is explained.

Afterward, a short introduction, how such networks are trained is given, which challenges occur, and how to tackle them. The importance of splitting the data into training and test set is also explained. Finally, it is described, which SotA approaches are used for single frame and sequence-based object detection.

2.2.1 Convolutional Neural Networks (CNNs)

Initially, a set of observed samples (X, Y) is given. Each sample $(x, y) \in (X, Y)$ represents a corresponding pair of an input x and an output y, which is called label. Each y is assigned to x by human supervision. In the application of VSD, x is the recorded sequence of images, and the label y contains the information, if and where there is smoke in this sequence. (X, Y) are the recordings during all smoke and non smoke experiments.

The goal is to use the observed samples (X, Y) to find a function f, which generalizes to arbitrary data and finds the correct label y for an input $x \notin X$, i.e. $y = f(x)$. f usually is a complex non linear function with a completely unknown structure.

Neural Networks (NN) are one way to approximate f by a parameterized function. Let $\mathcal{N}_\beta$ be a NN parameterized by β. β should be determined by the observed data (X, Y), so that $\mathcal{N}_\beta = f$. Finding β is a typical fitting problem. A NN is a stack of layers, which consists of linearities and non-linearities (Figure 2.3).

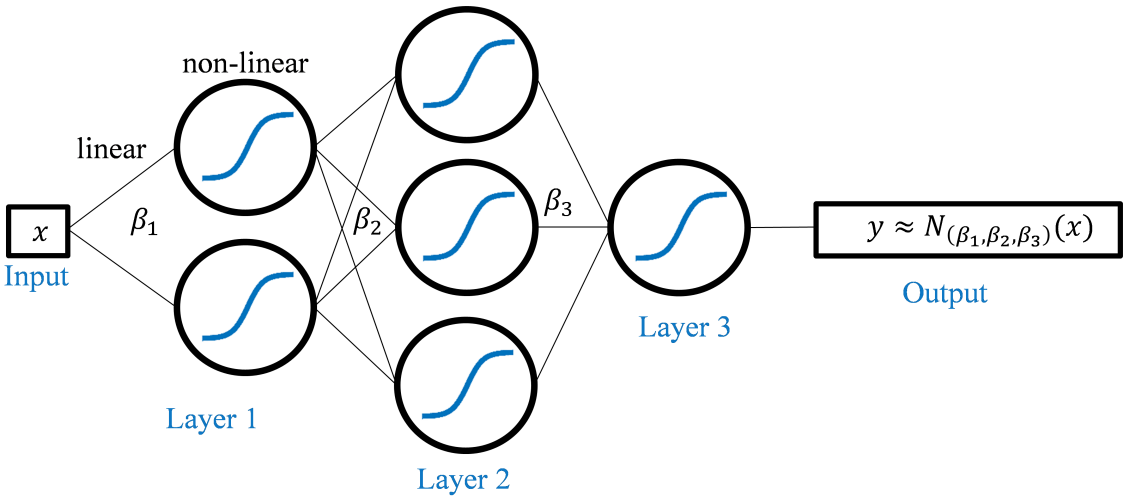


Figure 2.3: Schematic structure of a general NN. A NN propagates an input x through several linearities and non-linearities to predict a target output y. The parameters of a network are fitted to observations X with labels Y.

The challenge is that the NN generalizes from the observed data (X, Y) to arbitrary and unknown data $(\tilde{x}, \tilde{y}) \notin (X, Y)$. A NN has many degrees of freedom in architecture design, e.g. number of layers or number and kind of non-linearities of each layer.

A CNN is a special type of NN, which is usually applied to images. Common CNN

layers are convolutional, pooling and fully connected layers, which are described now.

A schematic illustration of a convolutional layer is given in 2.5. A convolutional layer consists of l convolutional kernels $K_i \in \mathbb{R}^{k \times k \times c}$ with $i = 1, \ldots, l$ and an activation function κ. $k \times k$ is the kernel size, typically 3×3, 5×5 or 7×7. c is the number of channels. The input of a neural network typically has three channels, one for each of the RGB input. Channels in intermediate layers of the neural network are the features extracted by the convolutional kernels from the input. Convolutional kernels can be interpeted as the common filter kernels used in image processing for discrete filters (Sobel, Laplace etc.). In contrast to classic feature extraction those kernels are parametrized and optmized for the sample data. The stride s is the distance between two centers of the widows of the tensor, which are convolved with K_i. κ typically is a nonlinear function, e.g. a sigmoid, tanh or a rectified linear unit ($ReLU$).

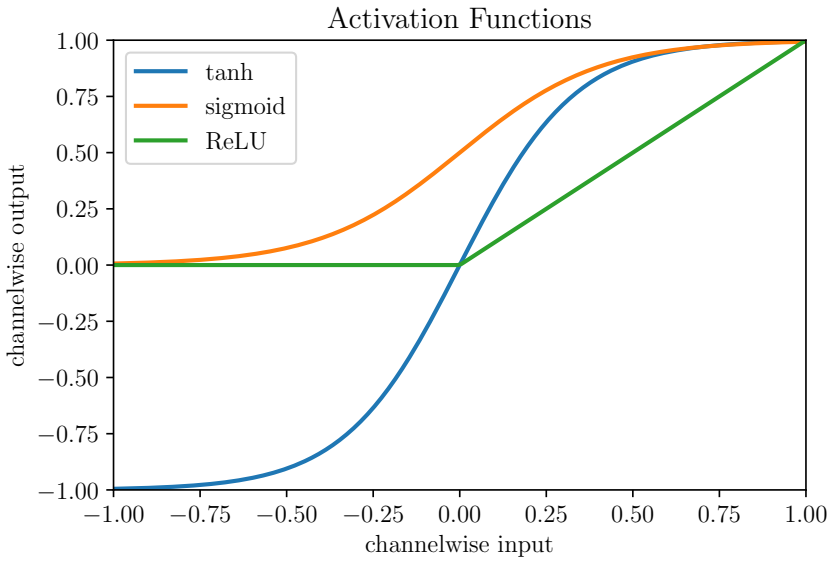


Figure 2.4: Most common activation functions.

Let $I \in \mathbb{R}^{h \times w \times c}$ be the input tensor of a convolutional layer (for the first layer is $I = x$ the input of the convolutional network). Then the output of a convolutional layer $O \in \mathbb{R}^{\lfloor h/s \rfloor \times \lfloor w/s \rfloor \times l}$ is

$$O_i = \kappa \left(K_i *_s I \right), \tag{2.2.1}$$

where $K_i *_s I$ is a discrete convolution with stride s. $h \times w$ is the spatial size of the input tensor (for the last layer is $O = y$ the output of the network). The choice of $s > 1$ is one way to reduce the output dimensions. The values of the kernels are all trainable.

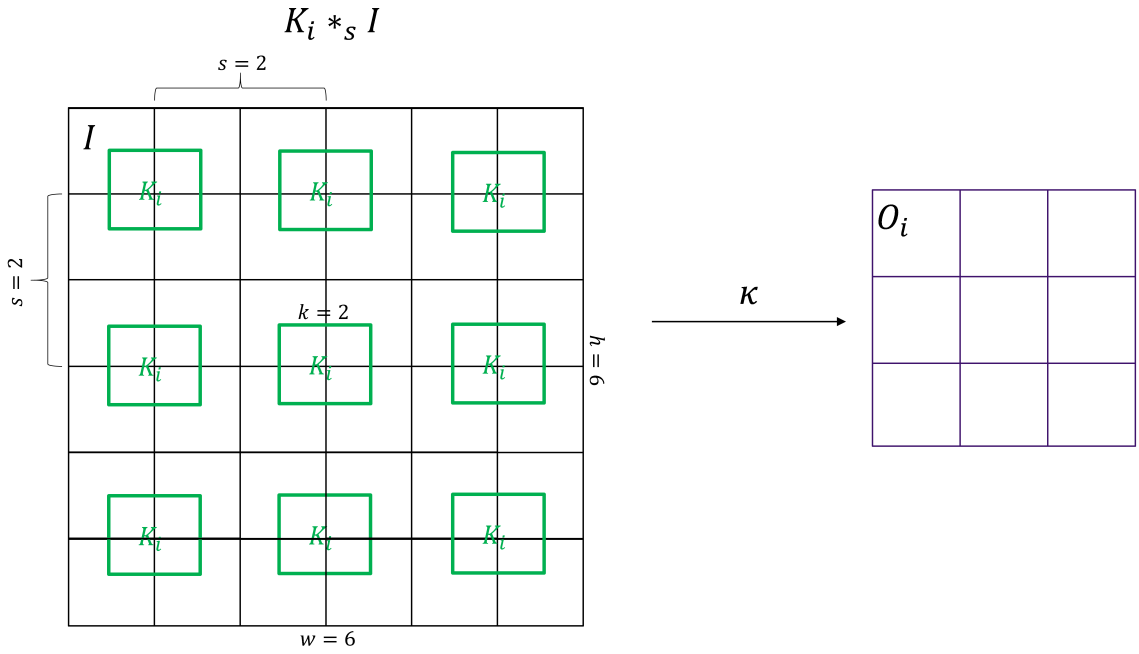


Figure 2.5: Components and parameters of a convolutional layer. The green rectangle represents a 2×2 convolutional kernel withs stride $s = 2$. Each black rectangle stands for one value in the $6 \times 6 \times 1$ input I. The convolution results in a 3×3 output O.

Pooling reduces the size of the input tensor by merging the information within a small tensor region. The most common pooling types are average and maximum pooling. Pooling layers have a pooling size $k \times k$, which is the region of which the maximum/average is taken, and they use a stride s, which is the distance in each dimension between the pooling centers. Pooling layers are not trainable.

Fully connected (FC) layers consist of an activation function κ and of a matrix $D \in \mathbb{R}^{l \times c}$. They transform the feature dimension by a matrix multiplication $O = \kappa(DI)$. The values of D are trainable.

The intuition behind CNNs is that they automatically learn to extract simple features in the first layers and in deeper layers, abstract correlations between these simple features, which are scale and rotation invariant. Examples for simple features are edges, contrast, and color. Complex features are harder to interpret for humans like the topology or texture of objects. Figure 2.6 illustrates the intuition

behind a CNN for smoke.

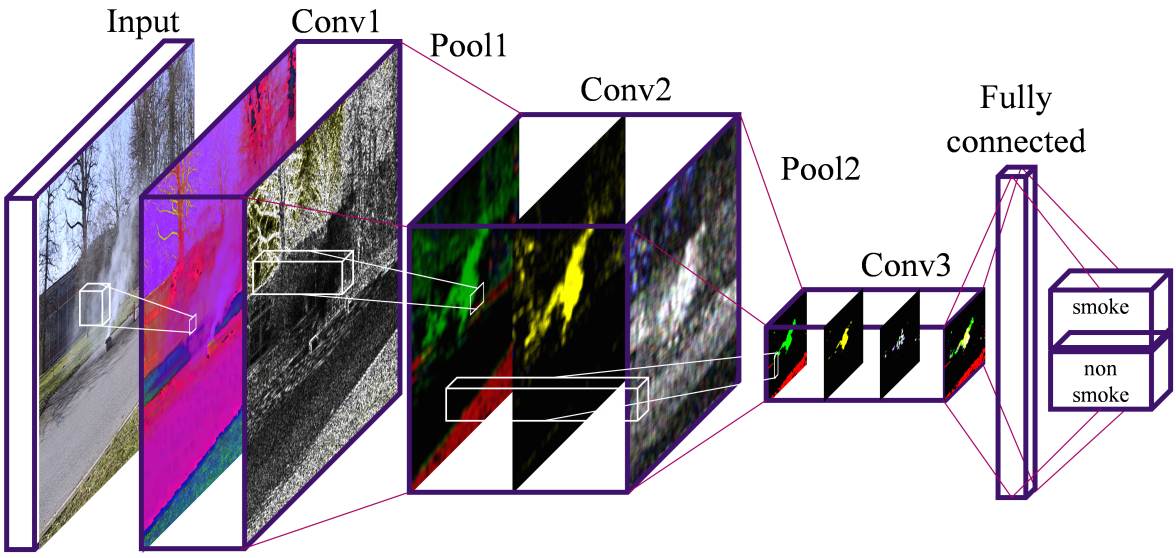


Figure 2.6: Illustration and intuition of a common CNN architecture. A CNN yields an image as input, and relevant information for the problem are extracted in each layer. Finally, the information is used to predict, if there is smoke or not.

Designing such networks is challenging. Since 2014 there has been exhaustive research in finding optimal architectures. The performance of architectures is usually benchmarked on *ImageNet* [32]. However, this is not the only indicator of the quality of a model architecture. Also, the number of parameter and calculation operations are taken into account to judge the ability to use such network in systems, which have hardware restrictions and real-time requirements.

Most popular architectural concepts are *VGG* [33], *ResNet* [34], *InceptionV1* [35] and *Xception* [36]. *VGG* uses the basic concept, which is increasing the features while reducing the spatial size. *ResNet* enables a much deeper network by just learning residuals while taking the number of features constant. Inception enables a network to learn, which kernel size is appropriate for each layer. *Xception* reduces the number of calculation operations by separately combining the features of a layer and afterward conducting depthwise convolutions. In this thesis, the *InceptionV1* is part of several DL methods, the single frame, the CNN+LSTM, and the CNN accumulation approach. For reduction of calculation complexity a network based on *Xception* is analyzed for VSD, the *MobileNetV2* [37].

2.2.2 3D-Convolutional Neural Networks (3D-CNNs)

3D-CNNs are highly correlated to (2D) CNNs the only difference is, that the input tensor $I \in \mathbb{R}^{t \times h \times w \times c}$, i.e. it has next to spatial size $h \times w$ and the channels c, a temporal dimension t. 3D-CNNs are supposed to extract and merge features in temporal and spatial domain, such that for the kernel and the pooling layers a further temporal dimension is added (Figure 2.7).

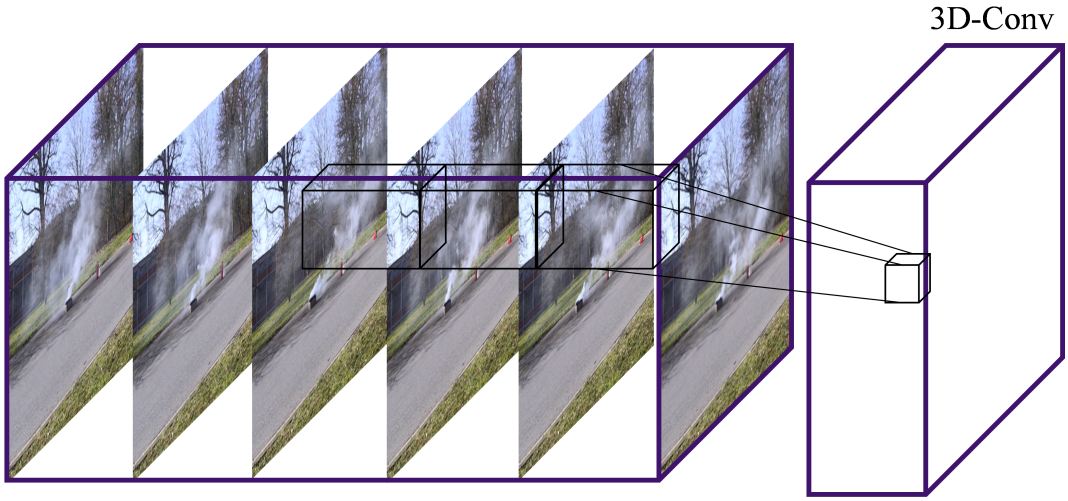


Figure 2.7: Illustration of a 3D convolution. The input of 3D convolutions is not only one image, but a sequence of images. 3D convolutions walk through the spatial and temporal dimensions.

The intuition behind these 3D-CNNs is that in addition to the spatial information, which can be obtained by an ordinary CNN, also temporal information can be learned, e.g., differences or optical flow in the first layers and object correlated changes or higher-order dynamic textures in deeper layers. Architectural research for 3D-CNNs is still at the beginning. In a pioneering work concerning temporal DL architectures it is shown that an inflated Inception architecture, namely inflated 3D (*i3D*) [38], outperforms every other architecture with temporal components on the *Kinetics* [39, 40] dataset, which is the biggest dataset containing sequences and sequence-based labels. The *Kinetics* dataset differs from the VSD use case in the sense that it contains rather short sequences ($\leq$ 10s) extracted from movies and private recordings, but mainly no surveillance scenarios.

In this thesis, the *i3D* is the architecture of choice when investigating 3D-CNNs.

2.2.3 Recurrent Neural Networks (RNNs)

In contrast to 3D-CNNs, which extract features from a fixed time window, RNNs extract new features from the last features, the memory, and the current input. Firstly RNNs for sequences are observed, where each time-step is a feature vector $x =$

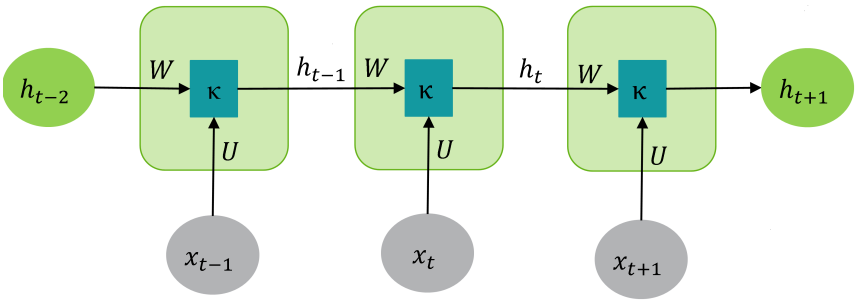


Figure 2.8: Schematic illustration of an RNN.

$(x_1, \ldots, x_T)$ with $x_i \in \mathbb{R}^c$. A simplified RNN (Figure 2.8) consists of an activation function κ and two weight matrices $U \in \mathbb{R}^{l \times c}$ and $W \in \mathbb{R}^{l \times l}$. The features $h_t \in \mathbb{R}^l$ at time-step t are calculated recursively

$$h_t = \kappa\left(Ux_t + Wh_{t-1}\right). \tag{2.2.2}$$

The values in U and W are trainable. Such basic RNNs show bad behavior during training of the values [41], since the gradient usually explodes or vanishes, when dealing with long sequences.

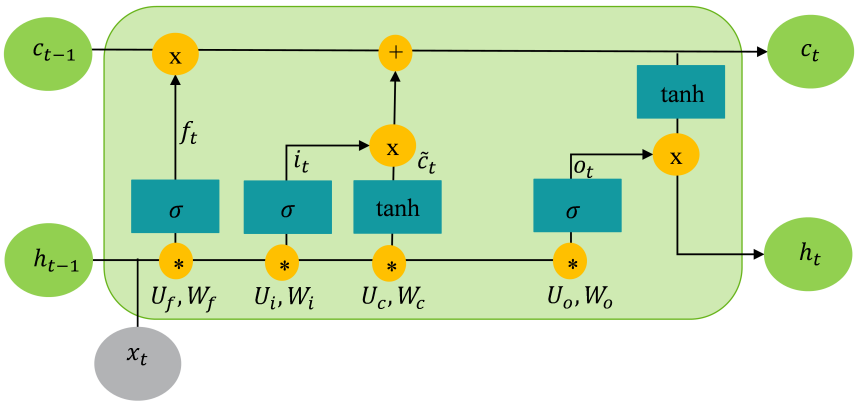


Figure 2.9: Schematic illustration of an (Conv)LSTM. For LSTMs the $*$ is a matrix multiplication and for ConvLSTMs it is a discrete convolution.

The RNN architecture which overcomes this issue is the long short term memory

network (LSTM), which is the architecture of choice, when using an RNN in research or application. The LSTM equations visualized in Figure 2.9 are as follows

$$f_t = \sigma\left(U_f x_t + W_f h_{t-1}\right) \tag{2.2.3}$$

$$\hat{c_t} = \tanh\left(U_c x_t + W_c h_{t-1}\right) \tag{2.2.4}$$

$$i_t = \sigma\left(U_i x_t + W_i h_{t-1}\right) \tag{2.2.5}$$

$$o_t = \sigma\left(U_o x_t + W_o h_{t-1}\right) \tag{2.2.6}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \hat{c_t} \tag{2.2.7}$$

$$h_t = o_t \cdot \tanh\left(c_t\right), \tag{2.2.8}$$

where σ is a sigmoid function, $\cdot$ an element-wise product and $U_\bullet, W_\bullet$ are the weight matrices. f_t is called forget gate, i_t input gate, o_t output gate.
c_t, the carry state, and h_t, the hidden state, are temporal information, which are transmitted to the next LSTM step: c_t carries the long term memory and h_t the short term memory. h_t is also the response of an LSTM block, which is for example used for classification or transfered to a higher layer.
Equation 2.2.3 determines, which ratio of the long term memory should be kept. Equation 2.2.4 can be interpreted as the impact of the input information on long term memory. The input information is given by 2.2.5 and 2.2.6 is the output information, which is scaled by the long term information in Equation 2.2.8 to yield the short term memory h_t. The critical component of the LSTM is Equation 2.2.7, in which the long term information c_t is calculated. The linear structure of this equation prohibits the gradients to vanish or explode during the optimization process.
To maintain the spatial structure of an input a straight forward extension of LSTMs is given by [42], the convolutional LSTMs (ConvLSTM), in which the matrix multiplications are replaced by convolutions. In ordinary LSTMs the input and output are vectors and the kernels are matrices, whereas in ConvLSTMs the input, output and kernels are tensors like in CNNs, so that the temporal states h_t and c_t correspond to feature maps of image regions. In addition to the information, a 3D-CNN can learn, a ConvLSTM can extract long term information, like noise or slow-growing. Intuitively, this information is characteristic of smoke.

2.2.4 Receptive Field

The receptive field of a NN is the part of the sequence and the image region that is associated with a certain output. This is the maximal information a NN is able to use to determine the output. The size of such a receptive field gives a hint, which information a network can combine for a decision and is the first step to understand, what a network does. The bigger the spatial receptive field, the more context information could be used by a network to generate a smoke prediction. However, the NN is also more complex to train and needs a higher depth and more parameters. Each dimension (time, width, height) of the input can be independently considered so that one can restrict to a one-dimensional consideration. It is assumed that the input dimension has arbitrary elements, and a theoretical maximal size of the receptive field is determined. It is calculated recurrently through the layers of a network. For pooling and convolutional layers the receptive field is calculated as follows: Let k be the kernel/pooling size and s be the stride of the i-th layer, then the size of the receptive field r_i of this layer is

$$j_i = sj_{i-1} \tag{2.2.9}$$

$$r_i = r_{i-1} + (k-1)j_i, \tag{2.2.10}$$

with $r_0 = j_0 = 1$. j_i is called jump.
A recurrent layer has a receptive field of size ∞. Equation 2.2.10 is used to calculate the receptive fields of all proposed architectures.

2.2.5 Training Neural Networks

Let $X = \{x_1, \ldots, x_n\}$ be the input data and $Y = \{y_1, \ldots, y_n\}$ the labels. The goal is to determine β, such that the average loss

$$\mathcal{L}(\beta) = \frac{1}{n} \sum_{i=1}^{n} L\left(\mathcal{N}_\beta(x_i), y_i\right) \tag{2.2.11}$$

is minimal. L is the loss function, e.g. the euclidian distance or the relative entropy. The optimization is usually done iteratively by gradient descent. Choose the initial parameter set β^0 according an arbitrary rule. Let β^i be the parameter set after the

i-th iteration. The next parameter set is calculated by

$$\beta^{i+1} = \beta^i - \rho \nabla_\beta \mathcal{L}(\beta^i), \tag{2.2.12}$$

where ρ is the learning rate and ∇_β the gradient operator. Equation 2.2.12 is repeated until β^i converges.

The learning rate is one of the most important hyperparameters while training a neural network. It significantly influences the convergence speed, if the algorithm converges at all and the quality of the minimum: If the learning rate is too low, one could end up in a local minimum, and if the learning rate is too big, the global minimum could be skipped over.

This iterative optimization process is called training, since the algorithm is improved step by step. X is called training data or training set.

A challenging aspect is to approximate the gradient $\nabla_\beta \mathcal{L}(\beta^t)$, since $\mathcal{L}(\beta)$ is a very complex function. This approximation is made recursively by backpropagation (BP), when training CNNs, and by backpropagation through time (BPTT) for RNNs. The idea is to update weights starting at the last layers using the chain rule.

A common way to train a NN is not updating the weights using the whole data set X at once, but divide X in b disjoint subsets $X_1, \ldots, X_b$, so-called batches. The weights are updated due to this batches, i.e., instead of one big update, b small updates are conducted. Doing all this b small updates is called an epoch. This process is named mini-batch optimization.

There are several extensions of this optimization algorithm to make it more robust and converge faster, e.g., introducing a momentum $\mu \in (0, 1)$

$$g^{i+1} = \mu g^i + (1 - \mu) \nabla_\beta \mathcal{L}(\beta^i) \tag{2.2.13}$$

$$\beta^{i+1} = \beta^i - \rho g^{i+1}. \tag{2.2.14}$$

The idea of momentum is that the gradients of consecutive batches are averaged and therefore the optimization process is more robust against the influence of local minima.

If the batches are constructed randomly new in each epoch, the optimization process is called stochastic gradient descent (SGD). SGD allows a good generalization from

training data to unseen data. Therefore all successful DL approaches in literature use a variant of SGD for training. In this thesis, SGD with momentum is applied.

2.2.6 Transfer Learning

Weight initialization is very important for training NNs. The challenge is to prohibit vanishing and exploding gradients or dead activations. One way to initialize the weights of a NN is to use probability numbers, e.g., the method of Glorot [43]. Since there is no external knowledge put into the weights, it is said that the network is trained from scratch.

In contrast to this approach, one can use transfer learning, which showed outstanding results in computer vision task, when dealing with small datasets [44]: A model which is trained for a completely other task is fine-tuned for the target task. The idea is that the new task has similar features, which have to be slightly different interpreted. Furthermore, pretrained weights are usually better initialization than random numbers. In literature transfer learning is very successful, especially, when having small dataset for the target task.

In this thesis, the influence of transfer learning is also investigated.

2.2.7 Overfitting and Split into Training and Test Set

During the optimization process, ML/DL models can learn to perfectly separate smoke from negative events in the training set, but this does not say anything about the performance of the algorithm on unseen data. The question is rather, how does the trained model generalize on unseen events. Therefore next to the training set a second set containing smoke and negative events has to be defined, the test set. The case that the decision function perfectly fits the training, but not the test set, is called overfitting, Figure 2.10. Especially complex models suffer from overfitting, also known as curse of dimensionality. But these models are necessary if the feature domain is very complex like for DL.

In practice, the model with the best performance on the test set is chosen, which is a good compromise between performance and generalization. The more independent the training and the test set are, the more representative are inferences from the test performance. The training data should also be representative to get a good gener-

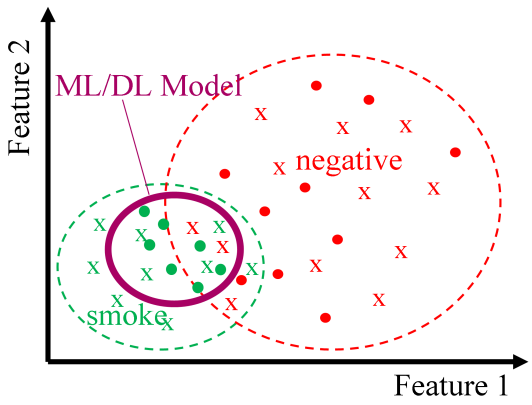


Figure 2.10: An illustration of overfitting. The dashed circles are all real-world smoke and negative data. The dots are training and the crosses test samples. Within the purple region are all the data, which the model predicts as smoke.

alization during the training phase. In Figure 2.10 for example the smoke training samples cluster in a small part of the whole set, which is a typical indicator for not being representative, whereas the test samples distribute equally in the whole set. Since different models are evaluated and compared on the test set, there is also an indirect optimization process on it. Therefore in practice, usually a third set is introduced, the validation set, which is used to measure overfitting effects during training. Finally, the test set is applied for the evaluation of the overall VSD algorithm. The validation set has to satisfy the same requirements as the training and test set. Nevertheless, the data used in this thesis are not comprehensive enough to construct three meaningful datasets covering the requirements. Therefore the sets are restricted to training and test, which are very carefully constructed to make meaningful inferences from the test set. This is also established practice in VSD literature, when DL methods are applied, e.g. [29, 30].

2.3 Object Detection and Sequence Classification

DL methods are increasingly successful applied in CV tasks. VSD is a combination of object detection and sequence classification since smoke can be treated as an object in a video sequence. So it is nearby to apply object detection as well as sequence classification methods to VSD.

In literature, object detection is usually done using DL methods for single frames, which is explained first. Application of DL methods to sequence classification is

a new research topic, not only in the field of VSD. Temporal DL methods, which are relevant for this thesis, are presented secondly. Finally, an overview, which DL strategies are applied to VSD so far, is given.

2.3.1 Single Frame Object Detection

Image classification and object detection are fields, in which DL methods immensely increase the performance and open new doors. A comprehensive overview of the developed techniques and overcome challenges is given by [45]. Object detection using DL methods is usually done by predicting a bounding box including the desired object in a single frame. The NN is trained according to two conditions: Firstly, the object has to be classified correctly. Secondly, the object bounding box has to be located as exactly as possible. In recent years there was a big progress in object detection. The best performance show two stage approaches [46–48]. In the first stage a *Region Proposal Networks* (*RPN*) predicts a rough region, where an object exists. The information in this region is afterwards compressed by ROI-pooling and fed to a second stage, which predicts the class of the objects and finetunes the boudning box.

In contrast one stage approaches like *You Only Look Once* (*YOLO*) [49] or *Single Shot Detection* (*SSD*) [50] predict class and bounding boxes at the same time. Therefore one stage approaches are easier to train and need less computational power, but also show worse performance.

In literature normally SotA CNN architectures like *VGG*, *Inception* or *ResNet* pretrained on *ImageNet* are used as so-called backbone networks to extract features. These backbone networks are combined with some post-processing afterward to obtain a full classification and localization pipe. *RPN* architectures usually give the best performance in object detection challenges, whereas *YOLO* or *SSD* allows a compromise between performance and resource cost. In this thesis, a slightly different approach is used. Therefore it is not necessary to go into details. Nevertheless, the basic idea remains the same: Extracting features by a backbone network and afterward classifying and localizing the smoke by a custom approach.

2.3.2 Sequence Classification

A rather new field of research is sequence-based classification, where complete image sequences are fed into a NN to describe scenes automatically. The idea is to combine spatial and temporal information appropriately.

Sequence classification tasks are not as comprehensive investigated as single frame tasks in CV, although a video sequence contains much more information than single frames. According to the author's estimation, there are mainly two reasons for this: Firstly, good quality video datasets are rare, secondly training temporal DL networks is very expensive in computing resources. E.g., the Google group used 60 GPUs for comparison of SotA temporal architectures in [38].

One way to overcome these computing resource issues is to extract temporal information by classic CV algorithms, e.g., optical flow, and use this as an input for a CNN. Using this approach, all stationary spatial information, like the topology of the recording environment, is discarded. Therefore some approaches combine optical flow with RGB information [51]. The disadvantage of this approach is that one is restricted to the temporal information of the few (usually two) frames used to calculate this temporal information.

Nevertheless, it exists some research in DL architectures, which can learn temporal information. Mainly two basic concepts became famous: Firstly extracting features by a CNN and utilize the resulting spatial feature vector of each frame as the input for a LSTM, secondly designing CNN like architectures using 3D-CNNs. Different architectures are compared by [38] on the *Kinetics* datasets, which consists of more than $300,000$ sequences and 400 action classes and has been available only since 2017. Despite the ability to automatically learn temporal information from RGB frames, the best results are obtained by giving the networks a priori temporal information like optical flow as an extra input. It turns out that an architecture, called *i3D*, which inflates the *InceptionV1* idea to a 3D-CNN, outperforms all other approaches. Furthermore, the authors showed, that pretraining on *Kinetics* also increased the performance on other temporal classification datasets and made the pretrained weights for *i3D* publicly available.

In this thesis, the CNN+LSTM and *i3D* approaches are applied to VSD and compared to each other.

Chapter 3

Data Preparation and Evaluation Concept

Data preparation is a crucial part of developing and evaluating a DL based algorithm since the algorithm can only learn what it sees during training. The higher the variation in the dataset, the better is the generalization to similar unseen situations. Data preparation consists of three parts: Gathering, structuring, and labeling.

Bosch has done gathering over several years. For this thesis, an exhaustive internal dataset is provided, which contains smoke recordings and non smoke surveillance scenarios. These are the raw data, which have to be prepared for developing and evaluating VSD algorithms. This chapter starts with the introduction of the *Bosch* dataset in Section 3.1.

Structuring includes cutting the raw data to events, clustering the events due to their properties, and splitting the data into training and test set. A structuring concept is presented in Section 3.2.

Labeling is to annotate the events, where smoke appears. Due to the effort, one can spend, labeling could be done in different granularities: From frame-wise, over bounding box to pixel-wise labels. With an acceptable effort, the smoke events in the training set are bounding box labeled. The sequences in the test set are frame-wise labeled. This is proposed in Section 3.3.

The goal of this thesis is firstly to verify that DL based algorithms can solve the VSD problem and secondly to find the DL approach, which is most suitable for VSD. The test sequences are used to compare different VSD algorithms for this purpose.

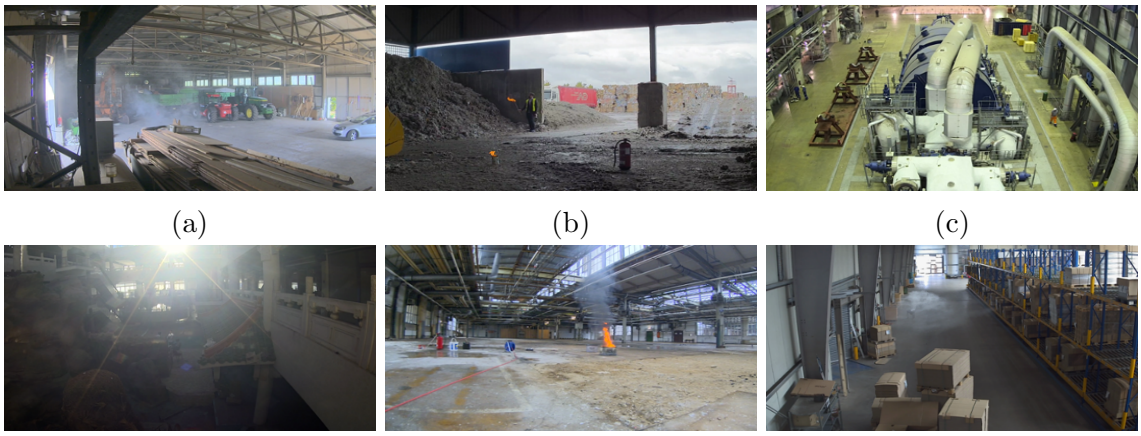
In Section 3.4, a measure is developed to evaluate the algorithm performance on the test set due to frame-wise labels. This measure represents the demands of a VSD algorithm to be fast and reliable.
Finally a VSD algorithm based on classic CV methods is evaluated using this measure in Section 3.5. This evaluation acts as a baseline, which is compared to the DL approaches analyzed in the next chapters.

3.1 The Bosch Dataset

DL requires a huge amount of data to cover all scenarios an algorithm is supposed to manage. The goal of VSD is to distinct smoke from all other events, which could appear in a surveillance scenario, so-called negative events. For this thesis, *Bosch* provides a huge amount of less structured smoke, and negative recordings gathered over the last years. The *Bosch* dataset consists of $1,600$ recordings, of which 630 contain smoke. The duration varies from one minute to about eight hours, in total 357 hours of video material. The recordings were taken in 82 different locations. At 56 of them, smoke experiments are conducted.

3.1.1 Smoke Recordings

In the last 8 years, *Bosch* conducted smoke experiments at different locations. Several differently positioned cameras record these smoke experiments. Some experiments are inspired by the tests, which are conducted to certify ordinary smoke detectors, which mainly concerns the combustible material, e.g., wood (open flame, smoldering), liquids (ethanol, gasoline, decalin), cotton and paper. These combustible materials are ignited by a flame, such that an open fire occurs, or by a hot plate so that only smoldering smoke is visible. Figure 3.1 shows some examples.
Experiments, according to standards, are costly. Therefore the most experiments are done with different colored and sized smoke cartridges, to simulate the visual behavior of real smoke events.



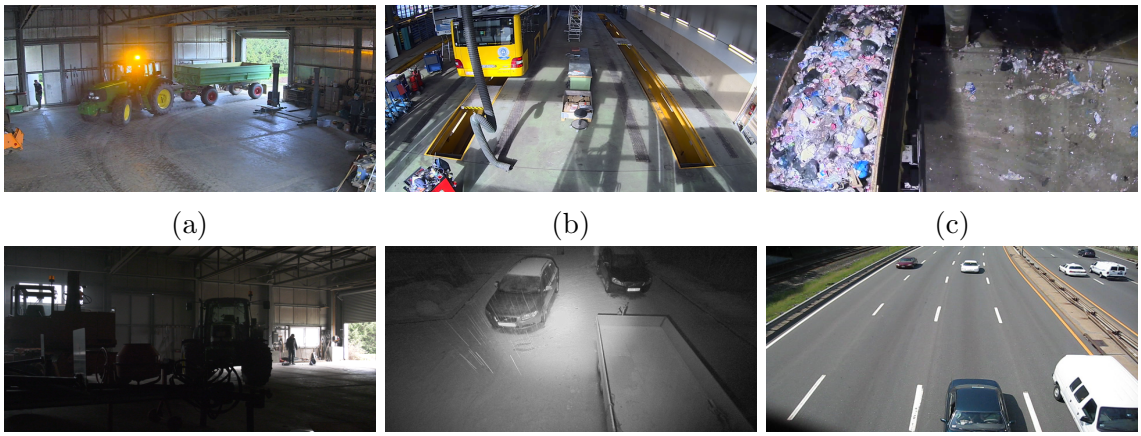
(a) (b) (c)

(d) (e) (f)

Figure 3.1: Smoke experiments at different locations and with different combustibles. (a) Smoke cartridge in a barn. (b) Open wood fire at a gravel plant. (c) Smoke cartridge in power station. (d) Smoke dust in a shopping center. (e) Liquid fire in a factory. (f) Smoke cartridge in a warehouse.

3.1.2 Negative Recordings

At one location, several experiments are conducted. Therefore there are multiple smoke events in one recording. Between these smoke, events are non smoke phases, in which negative events like opening a garage door, moving crane, persons or idle are recorded. Figure 3.2 shows some examples of negative recordings.



(a) (b) (c)

(d) (e) (f)

Figure 3.2: Some different kinds of negative recordings. (a) Tractor with flashing light in a barn. (b) Moving shadows while door opening in a bus station. (c) Emptying dustcart at a dumb. (d) Opening door in a barn. (e) Snowing at a parking area. (f) Passing cars on a highway.

These negative events can cause the VSD system to throw a false alarm. Periods without smoke between smoke events are not the only source for negative events

in the dataset. *Bosch* has a very comprehensive database consisting of various surveillance recordings. These recordings are also suitable to extract negatives events from it.

3.2 Data Structuring Concept

The goal of data structuring is to define a suitable training and test set. These datasets consist of smoke and negative samples. One sample is defined as a consecutive sequence of frames, which is called an event. It is not appropriate to use complete recordings as samples for training and testing since they are too long (up to several hours). Therefore meaningful smoke and negative events have to be extracted out of the recordings. Which rules are applied to extract such events is described firstly.
Each set has to contain a representative amount of events, which is assured by splitting the events due to characteristic properties, which is presented secondly.

3.2.1 Extraction of Events

For algorithm development and evaluation purposes, the raw data are divided into suitable smoke and negative events. The training and test sets are constructed of these events, whereas each event is one sample. Now the rules and guidelines for extracting smoke and negative events are defined.

Smoke Events

Smoke events are extracted from the recordings taken during fire experiments. In one recording, there are several smoke events. A smoke event starts without smoke for the first 5-10s. The idea is that algorithms using temporal information can attune to the background and distinguish it from smoke. In cases where smoke from the previous events is left, this rule can be ignored. A smoke event lasts from 30s to 500s. If an event is shorter than 30s, it is not considered. If an event is between 30s and 500s, it is cut, when there is no smoke visible anymore. If an event is longer than 500s, it is cut to 500s.

Negative Events

In general, every period of a recording without smoke in it is a candidate to be a negative event. Negative events also have a duration between 30s and 500s. Idle events, in which nothing happens, are uninteresting and not extracted as an event. If possible there are 5-10s of idle at the beginning and end of an interesting event. Interesting events are moving objects, blinking lights, or illumination changes.

General Rules

There are some general rules applied while extracting events. There should be no overlap between two arbitrary sequences, such that each frame is unique. Slight camera shaking is allowed because it is typical in environments, where big machines are applied. However, shaking due to camera adjustment should not occur in the events.

3.2.2 Description of Events

Smoke and negative events are described by properties, due to which it is ensured that each set contains representative events.
The events have a wide variety, which makes smoke detection challenging (Figure 3.3). The recordings are influenced by mainly three factors: Environmental conditions, recording conditions, and smoke types.

Environmental Conditions

Environmental conditions are dependent on the scenario, in which the smoke detection takes place: For example the location (indoor, outdoor, semi-outdoor), the weather (cloudy, sunny, rainy, snow), the illumination (artificial, natural) or air conditions (turbulent, windy, steady). Typical indoor applications for VSD are buildings with high ceilings ($>$ 5m), calm air conditions and artificial illumination. In such areas, smoke needs a long time to reach a detector, which is placed at the ceiling. For example a factory hall, a warehouse, a shopping center, a train station or an airport hangar.
In outdoor applications, no ordinary smoke detection is possible. Typical scenarios

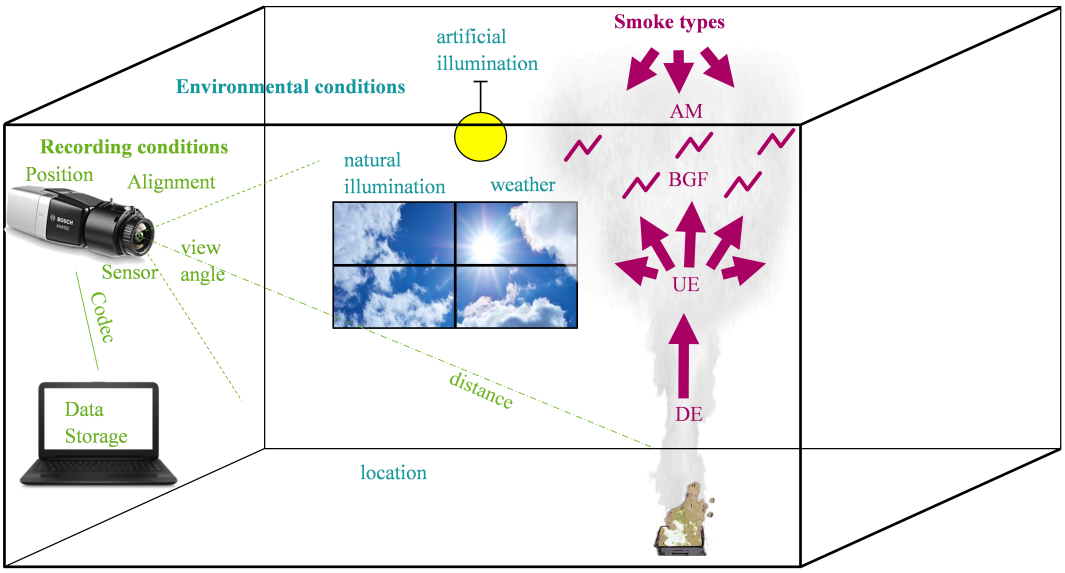


Figure 3.3: The smoke recording have a high variety. This requires a structuring and annotation concept. There are three main properties to describe events: recording conditions, environmental conditions and the smoke types (AM=Ambient Smoke, BGF=Backgorund Flickering Smoke, UE=Undirected Expanding Smoke, DE=Directed Expanding Smoke).

are a pedestrian area, a landing strip, a harbor or a garbage dumb.

Semi-outdoor applications are formally inside, but they have outdoor properties like turbulent air conditions or illumination changes by movement of clouds and sun. Examples are an underground/train station or a tunnel.

These influences determine the optical behavior of the smoke, the sensitivity, and flexibility of the smoke detection system.

Recording conditions

Recording conditions concern the camera and its installation. For example the position (bottom, middle, top), the alignment (viewing angle, opening angle), the sensor (type, resolution, framerate) or the preprocessing (tone-mapping, codec). The position and alignment impact the visual behavior of smoke. The sensor and the preprocessing deliver the raw data to be used for smoke detection. The noise disturbing smoke detection is also dependent on the sensor. In this thesis, it is assumed that the recording device is installed at a fixed position in a surveillance scenario. Slight shaking or camera movement is allowed, since it is typical for scenarios like in a production factory, where big machines are utilized. Restrictions on the sensor or the preprocessing are not defined.

Smoke and Negative Types

Smoke types describe the visual appearance of smoke, which includes temporal properties, e.g., the expanding or motion behavior, and static properties, like brightness, smoke density or texture. For VSD these properties are the basis to infer for unique features to distinct smoke from other events. A detailed discussion of smoke types is given in [52], where basically four smoke types are described: directed expanding (DE), undirected expanding (UE), background flickering (BGF) and ambient (AM) smoke. To each smoke event, there could be assigned at least one smoke type.
To negative events, the smoke types, to which they are similar is assigned. UE smoke has to be differed from slowly expanding objects like objects moving towards the camera or reflections when clouds are covering the sun.
Since thin BGF smoke is hard to distinguish from noise, noisy negative recordings could cause BGF algorithms to give false alarms. Additionally BGF smoke with higher smoke densities could be confused with reflections, shadows or illumination changes.
Ambient smoke is similar to all slow-moving and noisy events.

3.2.3 Training and Test Set

The split into training and test data is crucial for the development of VSD algorithms. The training set is used to optimize the model weights and the test set to compare the models. The optimal case would be, that the sets are independently sampled from the same representative distribution, formally: identical and independent distributed (IID). The raw data for this thesis does not allow to fulfill this requirement to the full extent: Even two events extracted from the same recording or two viewpoints of the same scene are highly correlated. Nevertheless one could reach an IID property for the training and test set to each other, which is sufficient to get meaningful results concerning the generalization ability of the model.
For the same representative distribution, it is required, that the sets have to consist of smoke and negative events, which cover all recording conditions, environmental conditions, and smoke/negative types equally. A good approximation of this is

reached if each set has a similar distribution of indoor, semi-outdoor and outdoor events and a similar distribution of smoke/negative types in smoke and negative events.

An approximation of independence is reached if each set contains events recorded in different scenes with different backgrounds, i.e., different environmental conditions. The sets have no overlap in recording locations to assure independence.

It is hard to get the same distribution in environments and smoke/negative types while prohibiting an overlap in locations if one aims not to omit some events. It is decided to keep all events and allow some deviations in distribution. Table 3.1 gives an overview of the resulting sets and Table 3.2 shows the property distribution. One can verify that the properties are adequately represented in each set.

Set	**Smoke**	**Negative**	**Total**	**Locations**
Training	704	879	1583	48
Test	312	362	674	37

Table 3.1: Number of events and different locations in training and test split.

Set	**Indoor**	**Outdoor**	**Semi-Outdoor**	**DE**	**UE**	**BGFAM**	
Training Smoke	65%	18%	17%	70%	46%	93%	40%
Test Smoke	50%	29%	21%	65%	51%	95%	33%
Training Negative	59%	20%	21%	30%	38%	95%	12%
Test Negative	50%	27%	23%	32%	34%	92%	15%

Table 3.2: Distribution of environments (indoor, outdoor, semi-outdoor) and smoke types within the sets.

3.3 Labeling of Smoke Events

Labeling is the most important step to define what a model has to learn. There are different methods to label objects in an event. These methods differ in accuracy and effort. Two classes should be distinguished: smoke and negative. Therefore only smoke events have to be labeled. Firstly, the different methods are discussed and it is explained, which is chosen for this thesis.

Labeling smoke exactly is more challenging than other objects like people or cars. Secondly, this and possible impacts are discussed.

3.3.1 Label Method for Smoke Events

Four granularities of labeling are possible. The roughest is to label the whole sequence if there is smoke in it or not. Sequence-wise labeling is initially given after separating the recordings into events. The next step is to label each frame if there is smoke in it or not. Frame-wise labeling is also done while separating the recording into events.

The sequence and frame-wise labeling does not take into account where smoke is located in the image. The location can be marked by drawing a polygon around the smoke region. In object detection, usually a rectangle is chosen, a so-called bounding box (BB). One benefit of using BBs to label objects in sequences is that the label process can be speed up by interpolation: One identifies two frames between the object moves or grows with constant speed and interpolates the bounding boxes linearly between those two frames. Interpolation saves a lot of time during labeling. The finest granularity of labeling would be pixel-wise labels, which means that each pixel containing smoke is labeled. Pixel-wise labeling is very time-consuming. Figure 3.4 shows examples for the different label methods.

(a) (b) (c)

Figure 3.4: Methods of labeling. (a) Example of frame-wise labels. Each frame yields a label: smoke or negative. (b) Example for BB label. A rectangle surrounds the smoke region. (c) Example of pixel-wise label. Each pixel containing smoke is labeled.

The more accurate the labels are, the better the learning problem is defined. Due to time restrictions, it was decided to label the 705 smoke events in the training set with BB, which is done in about 400 hours. Based on this, the training problem is defined. For the test set, there are frame resp. sequence-wise labels. This type

of labeling is sufficient to determine the evaluation measure, which is proposed in Section 3.4.

3.3.2 Smoke Specific Label Problems

Labeling smoke leads to ambiguities. If two different persons label the same smoke event with BB, the result is likely different. The reason for this is that the smoke density is not uniformly distributed within the smoke region [53]. In the boundary regions of a smoke plume, thin smoke appears, which can only be recognized due to plausibility considerations comparing with the last frames. This is illustrated in Figure 3.5.

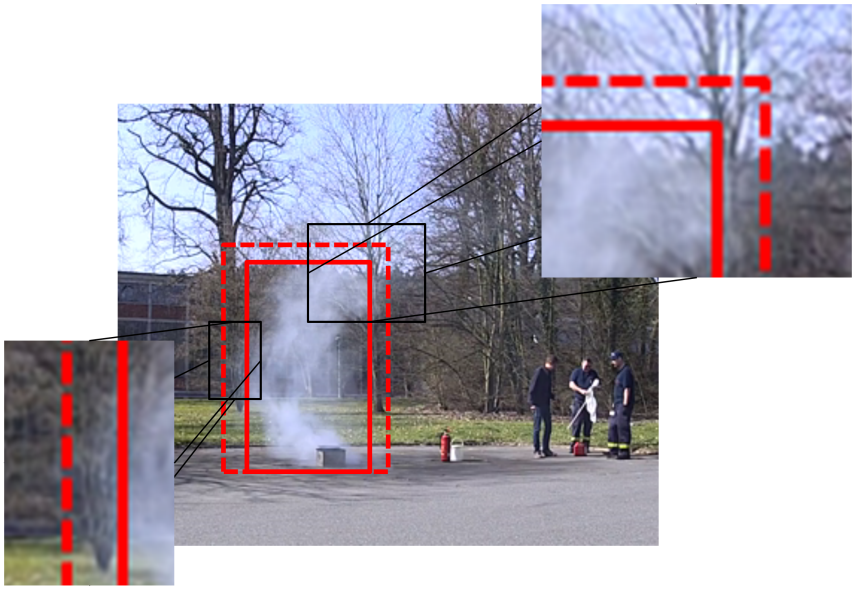


Figure 3.5: The label problem: Labels made by two different humans. The continuous rectangle focuses on smoke with a high density. Whereas the dashed rectangle tries to surround even thin smoke, which is hard to identify.

These labeling inconsistencies lead to contradictions when training a network to localize the smoke, which prohibits the best generalization.

3.4 Evaluation Measure

In the previous Section, a concept to prepare data for VSD algorithm development is presented. Now the question is, how the performance of VSD algorithms on the test set can be monitored. In this thesis, the performance of a VSD algorithm is

qualitatively defined as the reliability to detect smoke events as fast as possible while being robust against negative events. The goal of this Section is to transfer this qualitative definition into a concept to analyze VSD algorithms quantitatively. This quantitative analysis is finally compressed into one performance measure, the *Detection Speed*.

Since for the VSD purpose, low false-positive rates (FPR) are relevant, a receiver operating curve (ROC) is introduced, which is restricted to this low FPR. The area under curve (AUC) is used to measure the quality of the ROC.

The restricted ROC is extended, to assess the speed of detecting smoke. The algorithm only has a certain time, after the smoke is visible, to detect it, the detection time. For each detection time an ROC curve is determined. The *Detection Speed* is finally defined as the mean AUC of these time-dependent restricted ROCs. Since the exact location of smoke in a frame is not relevant, frame-wise labels are enough to evaluate VSD algorithm using the *Detection Speed*.

3.4.1 The Restricted ROC

The restricted ROC is introduced to assess the ability of an algorithm to detect smoke in a sequence while being robust against false alarms. Let n be the number of smoke and m the number of negative events. The false alarm and detection rate resp. FPR and TPR are defined as

$$\text{FPR} = \frac{1}{m}\#\text{false alarm events}, \tag{3.4.1}$$

$$\text{TPR} = \frac{1}{n}\#\text{detected smoke events}. \tag{3.4.2}$$

The VSD algorithm is assumed to be adjustable by a sensitivity parameter $\lambda \in [0, 1]$, which can be interpreted as the smoke probability threshold to throw an alarm. Increasing sensitivity leads to an increase of TPR and FPR. The ROC is the parameterized curve $(\text{FPR}(\lambda)/\text{TPR}(\lambda))$. Since only low FPR are relevant for VSD, the ROC is restricted to a FPR$\leq \theta$, where $\theta \in [0, 1]$ is the maximal feasible false alarm rate. Figure 3.6 shows the relation between the ROC and restricted ROC.

From these curves one can get an impression of the performance: The nearer the ROC curve is to the top left corner, the better the algorithm.

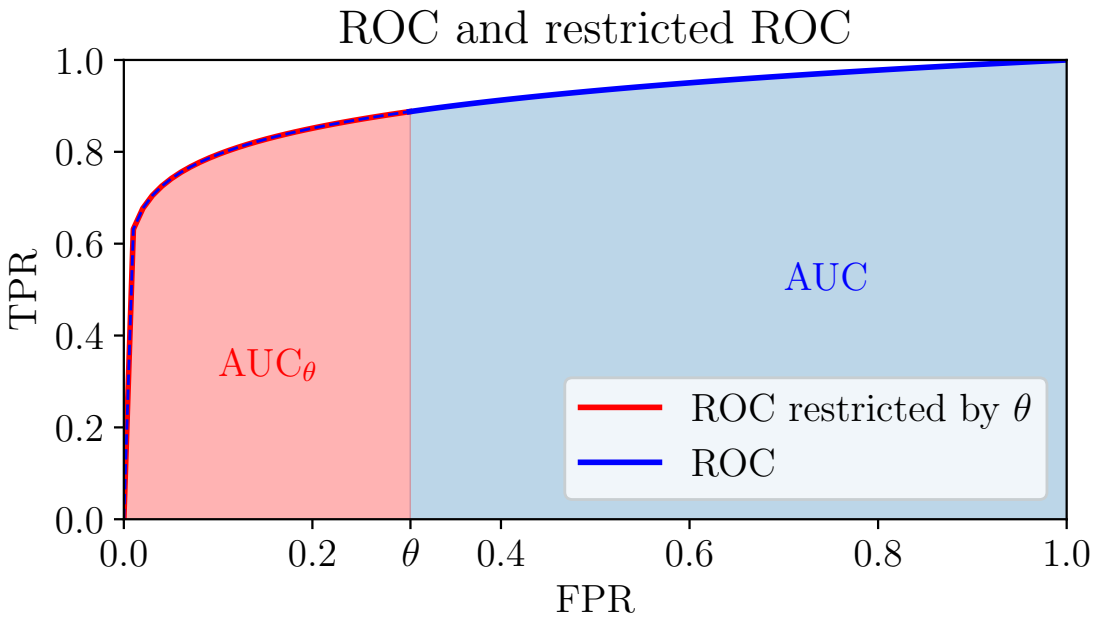


Figure 3.6: Relation between the ROC and the restricted ROC. The AUC and AUC$_\theta$ are used to measure the quality of the model, which generates this (restricted) ROC.

A typical measure to compare ROCs of different algorithms is the area under curve AUC.

$$\text{AUC} = \int_0^1 \text{TPR}\, d\,\text{FPR}. \tag{3.4.3}$$

For a complete ROC the maximum AUC is 1 and an AUC$<$ 0.5 is worse than guessing, if smoke is in a sequence. The area under curve AUC$_\theta$ for the restricted ROC is defined as

$$\text{AUC}_\theta = \frac{1}{\theta} \int_0^\theta \text{TPR}\, d\,\text{FPR}. \tag{3.4.4}$$

The factor $\frac{1}{\theta}$ normalizes the maximum of AUC$_\theta$ to 1. Note that guessing would lead to AUC$_\theta = \theta/2$.

3.4.2 Detection Speed

The restricted ROC is a quantitative concept to analyze the reliability to detect smoke while being robust against false alarms. But currently only complete events are considered, i.e. the restricted ROC does not contain any information about, how fast a smoke event is detected. Now a custom concept is developed to combine

the restricted ROC and the speed in one concept. The resulting measure is called *Detection Speed* D_{Speed}. It is constructed as follows:

1. Define a maximal false alarm rate θ and maximal detection time $T_{\max}$, at which the algorithm has to throw an alarm latest. The detection time is the duration from the start of the smoke event until it is detected. Note that in Section 1.2 the maximal detection time is set to $T_{\max} = 90\text{s}$.

2. Choose a fraction θ of all negative sequences for the algorithm, which are the most difficult, i.e., for which the algorithm has the highest smoke probability. In the next chapter, a classic CV algorithm is analyzed. Moreover, it is shown that $\theta = 0.02$ is suitable.

3. For the negative events use the whole sequence to calculate $\text{FPR}(\lambda)$. For each detection time t with $0 \leq t \leq T_{\max}$ consider only the period from the beginning t_{start} of the smoke event until $t_{\text{start}}+t$. This results in time-dependent $\text{TPR}_t(\lambda)$. Use this $\text{ROC}_\theta(t)$ to calculate time-dependent $\text{AUC}_\theta(t)$.

4. The *Detection Speed* is finally defined as $D_{\text{Speed}}(\theta) = \frac{1}{T_{\max}} \int_0^{T_{\max}} \text{AUC}_\theta(t)\text{d}t$.

Some corner cases are considered to get an intuition of this measure. $D_{\text{Speed}}(\theta) = 1$ means that every smoke event is detected without false alarms directly when it occurs. Whereas $D_{\text{Speed}}(\theta) = 0$ indicates that no smoke event is detected within $T_{\max}$ without throwing false alarms for all negative events. $D_{\text{Speed}}(\theta) = 0.5$ could have several meanings. Some examples are illustrated in 3.7.

3.5 Example Evaluation: Classic VSD Algorithm

Now a model-based VSD algorithm using classic CV methods is investigated. This investigation is an example of the proposed evaluation concept, and furthermore, a baseline for the later investigated DL-VSD algorithms.
The idea of the algorithm was developed by Stadler [54]. The algorithm concentrates on DE smoke. The basic property of DE smoke is a steady moving direction.
The algorithm is split into three parts. In the first part smoke candidate regions are

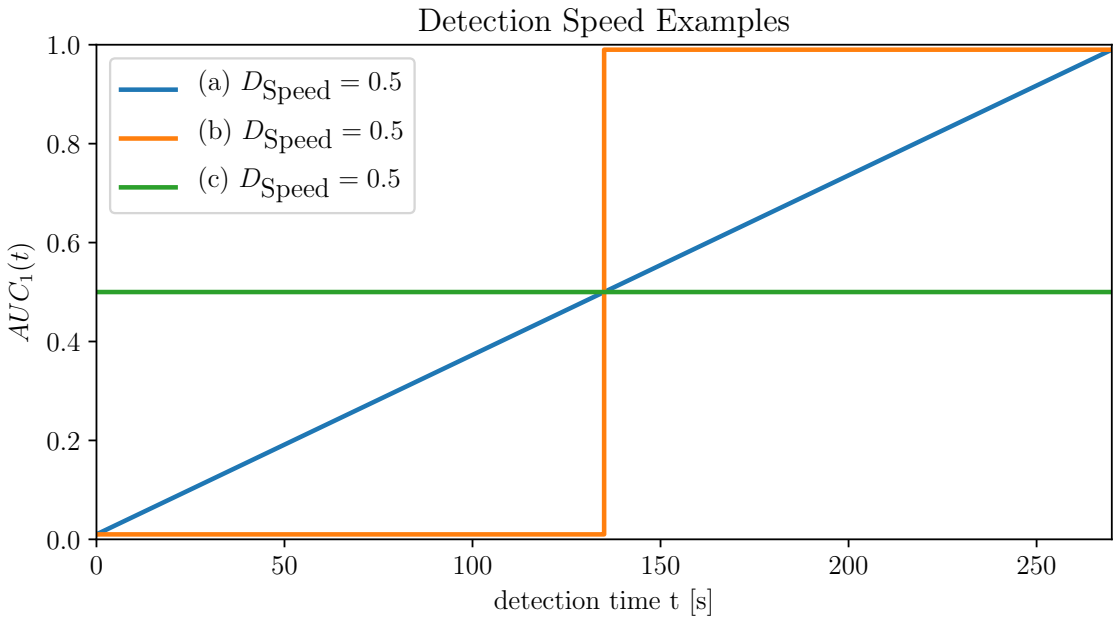


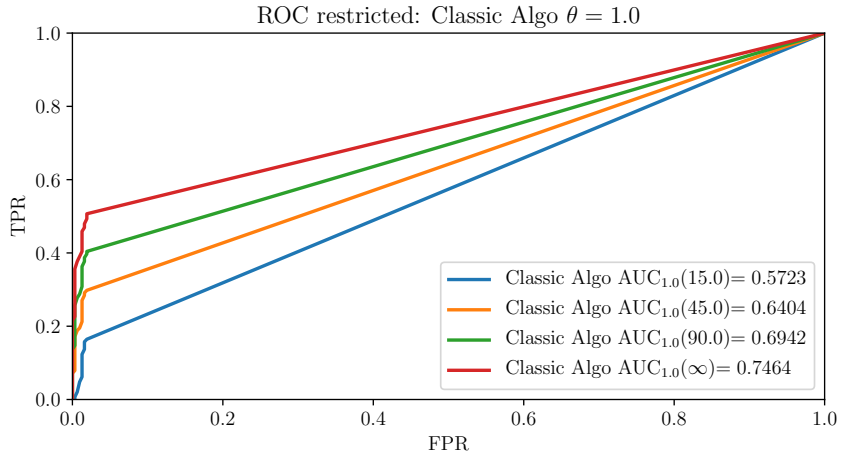
Figure 3.7: Some examples for $D_{\text{Speed}}(\theta) = 0.5$. T_{max} is set to $270s$ and θ to 1.0. (a) No smoke event is directly detected, when it occurs. The detection performance is linear increasing with the allowed detection time until all events are detected without false alarms after $270s$. (b) Within $t < 135s$ no event is detected without throwing a false alarm, but for a detection time of $t = 135$ every smoke event is detected without false alarm. (c) This is the typical for guessing (e.g. throwing a coin), if there is smoke in a frame or not. Note that the guessing line depends on θ. For $\theta = 0.02$ the guessing line is $\theta/2 = 0.01$

extracted by analyzing contrast and intensity changes over time.

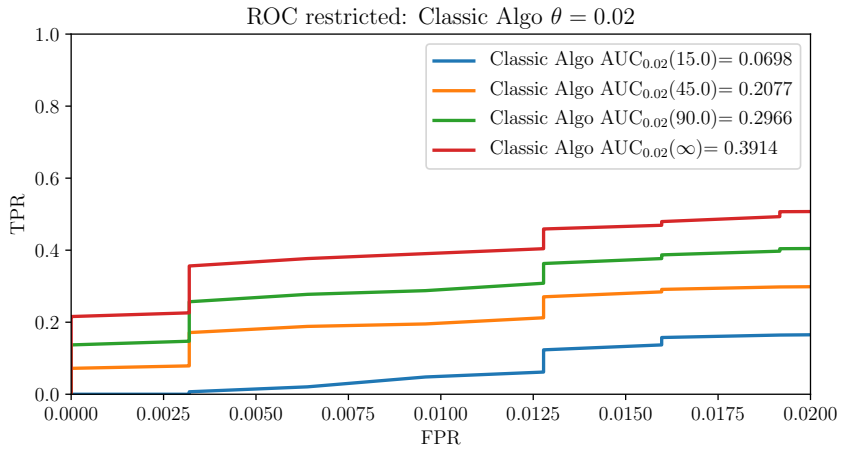
The second part is the key part. From an analysis of physical motion properties a smoke typical velocity range (1 px/s to 24 px/s) and motion angle range ($-90°$ to $+90°$ upwards) are inferred. The CV method of choice to measure such properties is the optical flow. If such motion properties are fulfilled over a sufficient time, the verification time, the third part of the algorithm is checked to give an alarm.

The idea of the third part is that the smoke source is nearly at a constant position over time. This is checked by the following condition: The velocity at the bottom of the event has to be at least 75% less than the mean velocity of the smoke candidate region.

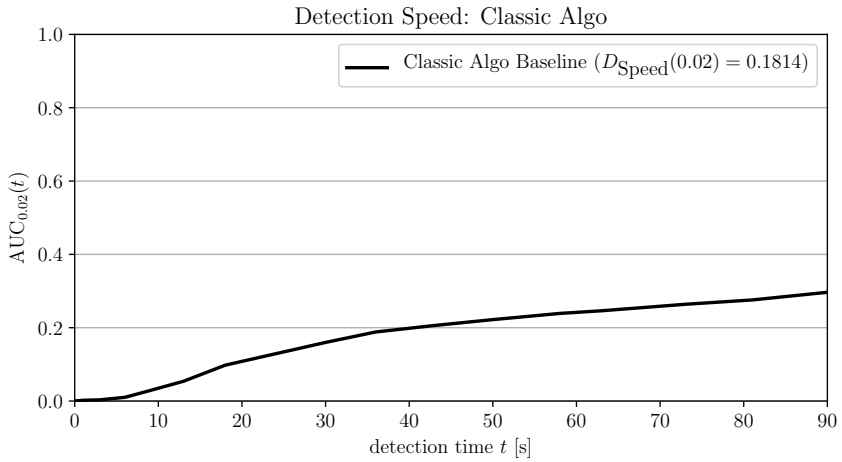
The sensitivity of the algorithm is controlled by the verification time: The lower the verification time, the higher the sensitivity of the algorithm. Higher sensitivity means that smoke events are detected more and faster, but also more false alarms are accepted. Note that the algorithm has been improved and adjusted many times



(a)



(b)



(c)

Figure 3.8: (a) shows the not restricted ROCs for the the detection times 15s, 45s and 90s. Also the resulting curve for an arbitrary detection time ($t = \infty$) is given. The figure is for $\theta = 1.0$, i.e. the full ROC without restriction. In (b) are the restricted ROCs for $\theta = 0.02$. (c) $AUC_{0.02}(t)$ curve and resulting *Detection Speed* $D_{\text{Speed}}(0.02)$. The maximum valid detection time is set to 90s.

to increase the performance. Discussing all details of the algorithm is beyond the scope of this thesis. The parameters of the proposed algorithm are adjusted by hand using sequences from the training set and evaluation is done on the test set.

Figure 3.8 shows that the classic algorithm is robust against false alarms, since the FPR for the lowest verification time is about 0.02. Also interesting: increasing the allowed detection time beyond 90s can improve the result significantly, but this is out of requirement.

The classic algorithm should be a baseline and compared to the DL approaches in the next chapter. Therefore it is suitable to restrict the ROC to $\theta = 0.02$ and use the *Detection Speed* $D_{\text{Speed}}(0.02)$ as performance measure in this thesis. The result shows that there is a big gap for improvement, which is tackled by DL approaches in the next chapters.

Chapter 4

DL-VSD Concept and Framework

In the previous chapter, a comprehensive dataset for training and test of ML/DL based VSD algorithm was described. This dataset is utilized to apply state of the art DL based algorithms for object detection to the field of VSD. In object detection algorithms are benchmarked in challenges on public datasets like *ImageNet* or *PascalVOC* [55]. These challenges consist of diverse detection problems with yearly increasing difficulty. Up to now, the challenges are single-frame detection problems, i.e., objects are detected and localized within one frame without using information from other frames. In VSD, the scenario is typically surveillance, and one has access to past information for a real-time decision. For an alarm, one can give the algorithm up to 90 seconds to decide, the faster, the better. Furthermore, not the whole smoke region has to be detected. For an alarm decision, it is sufficient to detect parts of the smoke region. These requirements are translated into a concept and framework for training and evaluation in this chapter.

Different DL-VSD approaches are compared. All these approaches have an underlying common framework and concept. All approaches utilize the same data, labels, training framework, and measures. Firstly, the data preprocessing is explained in Section 4.1. In Section 4.2, the custom cell wise classification approach, which is very suitable for VSD, is introduced. Furthermore, the model structure and the training problem, including the loss function, are defined in general. Augmentation and regularization increase the robustness of DL models and prohibits overfitting. These are described in Section 4.3. How the evaluation of DL-VSD approaches is

integrated into the framework is described in Section 4.4. The framework is implemented in *Python* using *Keras* and *Tensorflow*. Detailed implementation hints are given in Section 4.5. The proposed concept has many degrees of freedom, and some decisions are made due to plausibility considerations and experiences. The chapter closes with a discussion of those aspects in Section 4.6.

4.1 Data Preprocessing

Many decisions concerning the input are made for training the DL-VSD system. Varying the resolution during training could increase the robustness of the system, since it is a kind of data augmentation. However, after a few experiments, it is found that for comparable and reproducible results, all approaches require a similar temporal and spatial resolution for training and testing.

The raw data format varies in resolution and frame rate. The resolution reaches from 256×144 to 2992×1620 px^2 and the frame rate from 12.5 to 60.0 fps. The higher the resolution and framerate, the higher the computation resources and time is needed to train and evaluate a DL architecture. For simplicity, the original resolution and frame rate are normalized.

The resolution is normalized inspired by [38]: The frames are resized to $r_v \times r_h = 256 \times 256$. Especially for augmentation purposes, it has advantages to end up with the same width and height.

Now the goal is to adjust the frame rate. A compromise between computation complexity and relevant information has to be made. Therefore the smoke speed v_{image} in the plane of the 256×256 images is estimated according to [54]. Smoke has a velocity from $v_{\min} = 0.2$ m/s to $v_{\max} = 0.35$ m/s for test fires, which are commonly used for certification of smoke detectors. It follows that choosing an average speed of $v_{\text{mean}} = 0.275$ m/s is suitable. The most recordings have a vertical opening angle of about $\eta_v = 60°$. Assuming that the main component of the velocity is vertically directed and that the distance to the smoke source is about $d = 20$ m. The vertical

speed in image coordinates is calculated by

$$v_{\text{image}} = \frac{r_v/2}{\tan(\eta_v/2)} \frac{v_{\text{mean}}}{d}, \tag{4.1.1}$$

where the first fraction is the focal length. Using the assumptions above one yields $v_{\text{image}} = 3.1$ px/s. To see this relevant movement the frames are extracted by 3 fps using a constant frame skip.

The resulting sequences are stored as arrays with one byte for each channel in each pixel. A structured investigation of resolution and frame rate is not in focus of this thesis. Intuitively increasing the resolution and frame rate should give better results, since the information is more detailed. But there are two disadvantages, which could lead to worse results: Firstly, the temporal and/or spatial receptive field decreases, when using the same architecture. Secondly, hardware restrictions make it harder to train a network.

Nevertheless the chosen approach is a compromise and allows it to compare all DL architecture modalities.

4.2 General DL-VSD Methodology

Now the question is answered, what the DL architecture has to learn or how the VSD problem is defined for the training phase. Furthermore it is described, how the weights of a DL model are updated and finally, what augmentation is useful to prevent overfitting.

Now a short mathematical formulation of the initial situation is given. Let $S_1, \ldots, S_n$ the training sequences, where $n = 1583$ (704 smoke, 879 disturbance sequences). Every sequence S_i consists of T_i ($\overline{T} = 564$) frames, i.e.

$$S_i = (f_{i,1}, \ldots, f_{i,T_i}), \tag{4.2.2}$$

where $f_{i,t} \in \{0, \ldots, 255\}^{256 \times 256 \times c}$ is the t-th frame. c is the number of channels.

For each frame of each sequence $f_{i,t} \in S_i$ exists a bounding box label

$$B_{i,t} = \{x, \ldots, x + w - 1\} \times \{y, \ldots, y + h - 1\}, \tag{4.2.3}$$

where $(x, y) \in \{1, \ldots, 256\}^2$ is the upper left corner and $(w, h) \in \{1, \ldots, 256\}^2$ the width and height of the bounding box. If there is no smoke in the frame, it is set $B_{i,k} = \emptyset$.

4.2.1 Cell-Wise Classification

The VSD problem differs from other object detection problems, because it is not necessary to fit an accurate bounding box to the smoke region. It is sufficient to classify at least one part of the smoke region for sure. The idea is to map a binary label mask to each frame, which corresponds to the output size $o \times o$ of the DL architecture, which is quadratic for all investigated approaches.

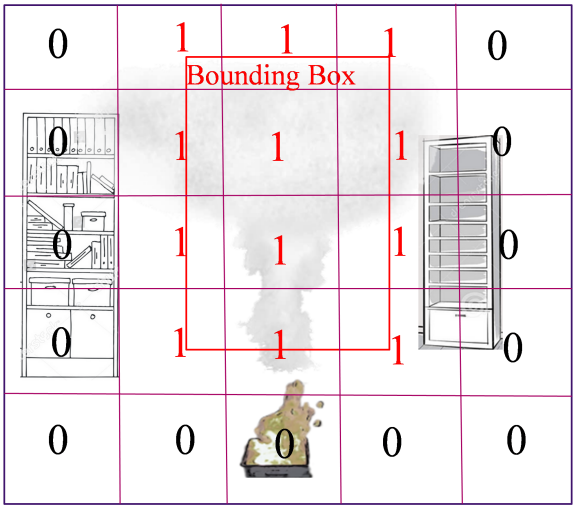


Figure 4.1: Cell-wise classification. A frame is separated into a grid. Each cell yields the label 1, if it intersects with a bounding box and 0 otherwise.

Each frame is covered by a grid with o^2 quadratic cells $C_{r,s}$ with $r, s = 0, \ldots, o-1$,

$$C_{r,s} = \{zr + 1, \ldots, z(r+1)\} \times \{zs + 1, \ldots, z(s+1)\}, \tag{4.2.4}$$

where z is the size of a cell. The cell size depends on the output size $o \times o$ of the DL architecture For example if the output of the architecture is 8×8, then $z = 32$. The binary mask $M_{i,t} \in \{0, 1\}^{o \times o}$ for each frame $f_{i,t}$ is defined as

$$M_{i,t}(r, s) = \begin{cases} 1, & \text{if } B_{i,t} \cap C_{r,s} \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{4.2.5}$$

That means, if a cell intersects with the bounding box the binary label mask is set to 1 and 0 otherwise (see Figure 4.1). One can interpret the binary mask as the probability of smoke existence.

4.2.2 DL-VSD Model Template

Now a template is proposed to design DL-VSD approaches, which meets the requirements and can easily be applied to a single frame and temporal methods.

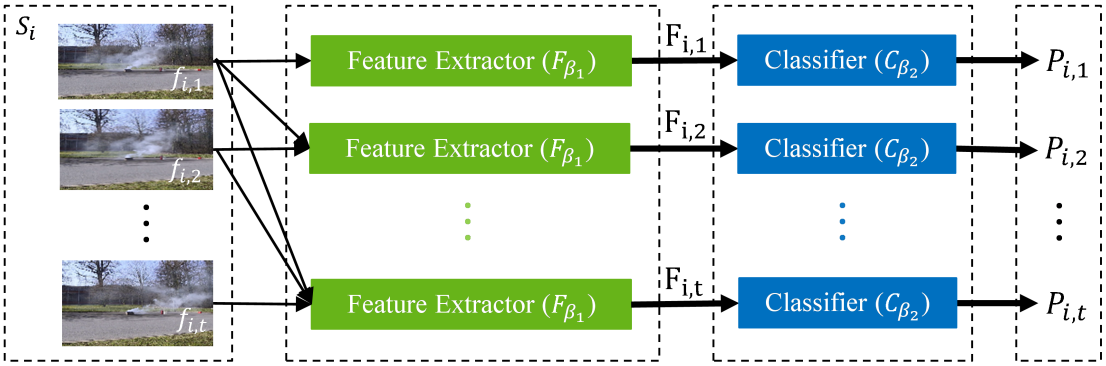


Figure 4.2: Overview of model template and information flow. Each DL-VSD model in this thesis is constructed of two parts: The feature extractor $\mathcal{F}_{\beta_1}$ and the classifier $\mathcal{C}_{\beta_2}$. To predict a smoke probability map $P_{i,t}$ for frame t in a sequence S_i. $\mathcal{F}_{\beta_1}$ extracts a feature map $F_{i,t}$ from the past frames $f_1, ..., f_t$. The feature map is propagated to classifier $\mathcal{C}_{\beta_2}$, which predicts the smoke probability map.

Let $\mathcal{A}_\beta$ be the DL system parametrized by β. β are the trainable parameter of the neural network, which are optimized with respect to the smoke detection problem. $\mathcal{A}_\beta$ is separated into two parts: the feature extractor $\mathcal{F}_{\beta_1}$ and the classifier $\mathcal{C}_{\beta_2}$

$$\mathcal{A}_\beta = \mathcal{C}_{\beta_2} \circ \mathcal{F}_{\beta_1}, \tag{4.2.6}$$

where $\beta = (\beta_1, \beta_2)$.

$\mathcal{F}_{\beta_1}$ extracts features from the past t frames of the sequence S_i and constructs a feature map $F_{i,t} \in \mathbb{R}^{o \times o \times l}$ with size $o \times o$, containing l features in each entry of the map

$$F_{i,t} = \mathcal{F}_{\beta_1}\left(f_{i,1}, \ldots, f_{i,t}\right). \tag{4.2.7}$$

Note for single frame architectures $\mathcal{F}_{\beta_1}\left(f_{i,1}, \ldots, f_{i,t}\right) = \mathcal{F}_{\beta_1}\left(f_{i,t}\right)$. In all proposed approaches the feature extractor is designed, such that a feature map is constructed

for each frame. For each cell the center of the receptive field is the center of the cell, so the features specially correspond to the cell.
The classifier $\mathcal{C}_{\beta_2}$ predicts smoke probabilities for each cell out of the feature map. The result is the smoke probability map $P_{i,t} \in [0,1]^{o \times o}$

$$P_{i,t} = \mathcal{C}_{\beta_2}\left(F_{i,t}\right). \tag{4.2.8}$$

To summarize this: The whole DL-VSD model $\mathcal{A}_\beta$ should be trained to predict a smoke probability map at time t according to all frames it has seen until a time point t

$$P_{i,t} = \mathcal{A}_\beta\left(f_{i,1}, \ldots, f_{i,t}\right). \tag{4.2.9}$$

The difference to other sequence based approaches is, that every frame yields a smoke prediction without using information from future frames. This is the typical real-time surveillance problem. Figure 4.2 gives an overview of the model template and the information flow.

4.2.3 Training Problem Formulation

According to the cell-wise classification approach the DL system is supposed to learn to predict a probability map $P_{i,t}$ of smoke for each time point t and each sequence S_i. The DL system is optimized, such that the mean loss between the prediction maps and binary label maps

$$\overline{L} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{T_i}\sum_{t=1}^{T_i} L(P_{i,t}, M_{i,t}) \tag{4.2.10}$$

is minimal. For $\mathcal{L}$ the mean over the binary crossentropy of each cell is used, i.e.

$$\begin{aligned}\mathcal{L}(P_{i,t}, M_{i,t}) = \frac{1}{o^2}\sum_{r=1}^{o}\sum_{s=1}^{o} & M_{i,t}(r,s)\log\left(P_{i,t}(r,s)\right) \\ & + \left(1 - M_{i,t}(r,s)\right)\log\left(1 - P_{i,t}(r,s)\right).\end{aligned} \tag{4.2.11}$$

The binary cross-entropy is a standard loss function for a two-class problem.

4.2.4 Optimization of Model Weights

Model weights optimization is usually done iteratively by gradient descent. One iteration over all samples is called epoch. A good concept to prevent overfitting is SGD (see Section 2.2.5). The idea is to update the weights not on all samples at the same time, but randomly decompose the training set into subsets, the batches. The number of samples within a batch is called batch size. For parallel processing on the GPU each sample of the batch must have the same sequence length, which is called batch length. Each batch is completely loaded to the GPU and weights are updated there. Therefore the batch size and length are limited by the GPU storage. The model architecture also influences the maximal valid batch size and length since all interim results of a forward pass are stored for the backward pass. Depending on the DL approach 100-500 frames can be handled by the GPU at once.
The batch size and length are varied due to the different approaches, but they are fixed during the training of each model.
If a sequence is longer than the batch length, it is cut, and if it is shorter than the batch length, it is repeated until the batch length is reached.
SGD with a momentum of 0.9 is used. Furthermore, a gradient clipping to a maximum of 2.0 is conducted to suppress exploding gradients, which especially could occur at first few updates. The learning rate decays each epoch exponentially. A initial learning rate $\rho_0 > 0$ and a decay rate $\delta \in (0, 1)$ are chosen. The learning rate in epoch e is $\rho_e = \rho_0 \delta^{e-1}$.

4.3 Improvement of Training and Generalization

This section has two goals: Firstly to improve the generalization for unseen data and the overall performance. Secondly, to stabilize the parameter update and accelerate the convergence during the training process.
Augmentation deals with the first goal. The training set is augmented with events, which are transformations of original sequences. The second goal is tackled by regularization, which increases the problem complexity.

4.3.1 Augmentation

Data augmentation is a method to increase existing data artificially and prevent overfitting. The idea is to change the data slightly, such that a human further detects the object. Typical methods are cropping, rotating, flipping, or adding noise to the original data. For the training process off all DL-VSD approaches, cropping, rotation, and flipping are applied. Temporal changes of thin smoke have a small but characteristic magnitude. Since even weak noise suppresses such changes, artificial noise is not applied.

Cropping

In all DL approaches, temporal and spatial cropping is used. Spatial cropping means that not the whole spatial part with size 256×256 is used, but each epoch a uniformly chosen random subpart of size 224×224. This cropping is inspired by [38]. Note that the same crop is used for each frame in one sample.
For temporal cropping a coherent subsequence $f_{i,t_1}, \ldots, f_{i,t_2}$, where $1 \leq t_1 \leq t_2 \leq T_i$. t_1 is randomly chosen according to a uniform distribution. The sequence length is $|t_2 - t_1| + 1$. This sequence length is fixed during training. If a sequence is too short, it is not cropped, but repeated until the sequence length is reached.
The main advantage of cropping is that it prevents overfitting since it is unlikely that the network sees the same sequences in multiple epochs.

Rotation

The sequence is either rotated by 90° to the left, the right or not rotated each with the same probability. Angles smaller than 90° can not be computed efficiently on the CPU and increases the training time a lot. Additional it requires padding and interpolation. Whereas a rotation by 90° only needs a resorting of the values.
The most smoke events have the DE (see Section 3.2.2) property, i.e., they move in one direction, mainly upwards or diagonal. Through rotation, the DL system is forced to focus on all directions, which is desirable since every smoke event should be detected.

Flip

With equal probability, the sequence is flipped at the vertical axis, horizontal axis, or not. Flipping is also very efficient to implement since it is again a resorting of values.

4.3.2 Regularization

Regularization stabilizes the training and can increase generalization to unseen data by making the training problem more complex. The methods applied during training of the models are explained now. All of them lead to better results and some of them to faster convergence.

Batch Normalization

Batch Normalization [56] is a concept to normalize the input of an activation function, which leads to a much faster convergence of the network. It is usually placed between a (3D-)convolutional layer and the activation function. The normalization parameters are determined for each batch independently. Let $\mu_{\mathcal{B}}$ be the mean and $\sigma_{\mathcal{B}}$ the standard deviation of the input X in batch $\mathcal{B}$. Then the input is normalized by

$$X_{\text{BatchNorm}} = \frac{X - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}. \tag{4.3.12}$$

This normalized input $X_{\text{BatchNorm}}$ has the mean 0 and the standard deviation 1. This simple transformation prohibits so-called dead activations. These are layers, in which the activation function of the input results in values and gradients near zero so that it does not have a significant influence on the weight update. Furthermore, it prohibits the output of a layer to be deterministic since $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are dependent on the batch, which is randomly constructed.

Dropout

Dropout [57] is a successful way to prevent overfitting. During each update of the parameters a random set of features is ignored. This means a random fraction $\tau \in (0, 1)$ of features is set to 0, which are called dropped features. They do not

influence the gradient during one update of network parameters. τ is fixed during the whole training process.
Dropping features prohibits deterministic outputs of layers since it is unlikely that in two distinct epochs the same features are used for the same sample.

Weight Regularization

Weight regularization is used to prevent extremely high weights by adding a penalty to the loss function $\mathcal{L}$. One common method is L2 regularization, which is applied in this thesis. Let W be the vector of all the weights in a model then the new loss is

$$\mathcal{L}_{\text{L2}} = \mathcal{L} + \omega||W||_2, \tag{4.3.13}$$

where $||\cdot||$ is the Euclidian length and ω the impact on the loss.

4.4 Testing

Evaluation of generalization and performance of different models is done on the test set. The goal is to find the model with the best results using the *Detection Speed* measure $D_{\text{Speed}}(0.02)$ (see Section 3.4). Each model is evaluated on the test set periodically during the training. Moreover, the best result is compared to the other models.
Note that for one evaluation of the test set, all the sequences have to be processed by the DL-VSD algorithm completely.

4.5 Implementation Details

The proposed DL-VSD framework and all models are realized in *Keras* [58] and *Tensorflow* [59]. The training is done on an *Nvidia TitanX*, which has 12GB memory.
One batch is constructed as a 5-D tensor: batch size, sequence length, height in, width in and channels. The output is designed as a 4-D tensor: batch size, sequence length, height out, and width out. Note that for single frame approaches the sequence length is set to 1.

The input of all models is normalized to [-1,1], since this is established practice in Literature. This normalization also matches all pretrained weights used in this thesis.

All layers of the network are wrapped by the *Keras* wrapper *Time-Distributed* except the temporal one, i.e., LSTMs and 3D-CNNs. As activation function always *ReLU* is chosen except for the LSTMs, for which the default Keras activations are used, i.e., tanh and hard-sigmoid.

Batch Normalization is placed between each (3D)-convolution and activation function. The normalization parameters μ and σ are updated during the whole training process by a running average, which is the Keras standard implementation.

Dropout is placed between the feature extractor and the classifier. In each sample, the same features are dropped, i.e., for all timesteps in one sample, the same features are chosen. The fraction of dropped features is $\tau = 0.4$. During testing, no features are dropped; they are scaled by the probability of being not dropped (0.6). Weight regularization is applied with ω=1e-5.

4.6 Discussion of Proposed Concept and Framework

The reader might notice that the proposed concept and framework has many degrees of freedom and hyperparameters, i.e., resolution and frame rate of the input, labeling, classification concept, loss function, learning rate (schedule), batch design, augmentation, validation loss, and regularization.

Since smoke normally covers 30%-80% of a bounding box and the label challenges (see Section 3.3.2), there are some cells labeled as smoke, without smoke in it and some cells not labeled as smoke with smoke in it (see Figure 4.1).

These "black sheeps" can lead to contradictions during training. The author suggests that removing such "black sheeps" is one of the most obvious levers for increasing the performance of DL-VSD systems.

For training BB information is transformed into cell-wise labels. It is also possible to predict the bounding box directly, which is usually done in object detection (c.f. 2.3.1). For this purpose, the classifier has to be adjusted.

When reflecting on the cell-wise classification concept, the question arises, why not use the sequence of one cell as a sample, which would be the usual object/scene classification approach in Literature. The critical point using the cell-wise classification concept is that it is similar to an ordinary classification concept, but with highly correlated samples in each batch. Because DL methods generalize best, when the samples are as independent as possible, this is a valid critic. However, the main advantage of the cell-wise approach is, that if the receptive field of the DL architecture is bigger than one cell, the contextual information of neighbor cells can be used for prediction of the smoke probability.

In this thesis, many hyperparameter combinations, regularization approaches, and further DL boosting techniques were tested and optimized concerning the author's experience and plausibility decisions. A structured analysis of all hyperparameters, for example, a grid search is to resource and time consuming and is therefore not in focus of this thesis. However, some compromises are made due to hardware restrictions. According to the author's estimation, it should always be possible to get slight improvements by changing some components in the concept or framework, but the biggest boost should always be available by increasing the data set and more accurate labels.

Chapter 5

Analysis of DL-VSD Methods

In the previous chapter, the general DL-VSD framework was introduced, which is applied to several DL approaches in this chapter.

The investigation starts with an application of the framework using a CNN, which only receives a single frame as input in Section 5.1. With RGB as input for the CNN, the training set is not comprehensive enough to get better results than the baseline proposed in Section 3.5. However, using transfer learning and weights pretrained on ImageNet the baseline is outperformed.

Single RGB frames only contain static information. For smoke, this information is not as characteristic as dynamic properties. Therefore it is assumed that temporal information increases the performance of DL-VSD models.

The simplest idea is to feed temporal information into a CNN. In Section 5.2 Flow and Diff are calculated from two consecutive frames and used as input for the CNN. It is shown that the performance significantly increases, whereas Diff input is by far the best.

Following the intuition that a smoke covered cell stays smoke covered over time, a handcrafted accumulation method is developed in Section 5.3, which extents the CNN by a low pass filter. This procedure improves the performance of the standalone CNN.

Using Diff or Flow as input, temporal information is extracted by handcrafted methods. In contrast, there are DL methods, which are designed to learn temporal information automatically.

The CNN+LSTM concept is described in Section 5.4. The idea is that the CNN

extracts spatial information. Afterward, this information is fed into a LSTM, which extracts temporal properties. It is shown that also for this approach Diff input performs better than RGB, with this procedure the performance of the CNN+Accumulation is again improved, especially in early smoke detection.
For all these methods containing a CNN, *InceptionV1* is applied.
Afterwards 3D-CNN is investigated in Section 5.5. The architecture of choice is the *i3D*, which reports the best results in a comprehensive temporal architecture comparison in literature (see Section 2.3). It is explained how the *i3D* is transformed to extract frame and cell-wise features. This architecture outperforms all the others, which qualifies it to be the best DL-VSD approach.
As classifier, a FC layer with sigmoid activation calculates the smoke probability for each cell independently in all approaches.
The findings are discussed in Section 5.6. All the analysis is done on the internal *Bosch* dataset. This prohibits comparison to SotA VSD algorithms. Therefore the analyzed DL-VSD approaches of this thesis are evaluated on public VSD datasets consisting of sequences in Section 5.7. Finally, to get an impression of the algorithm behavior in section 5.8 critical smoke and negative events are analyzed qualitatively.

5.1 Single Frame CNN for VSD

CNNs with single frame RGB input are the natural DL approach for VSD since they have been analyzed comprehensively and successfully applied to object detection tasks, also including VSD. The model template requires to define two parts (Figure 5.1): As feature extractor a successful SotA CNN is chosen: *InceptionV1* [60] (Figure 5.2).
The *InceptionV1* consists of repeatedly used 2D Inception blocks. The idea of these blocks is to extract information of different granularity in the parallel strands and combine them by concatenation. The stride 2 Max-Pool leads to a halving of the spatial dimensions. The output of an 256×256 frame f_t is an 8×8 feature map F_t with 1024 features each cell. This feature map is passed forward to the classifier frame by frame. As classifier, a FC layer with sigmoid activation is chosen, which

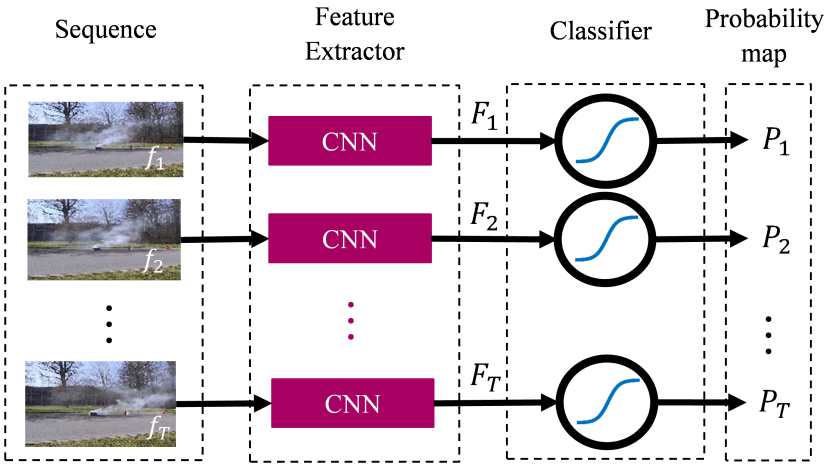
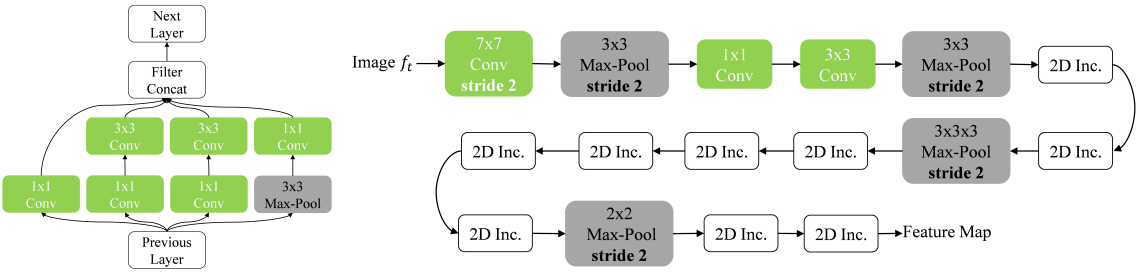


Figure 5.1: The arrows show the information flow of the single frame concept using a CNN. The frames $f_1, \ldots, f_T$ are independently passed through the CNN and features $F_1, \ldots, F_T$ are extracted. These feature are processed by a FC layer with sigmoid activation to calculate frame-wise probability maps $P_1, \ldots, P_T$ for smoke.



(a) (b)

Figure 5.2: The *InceptionV1*. (a) shows the repeatedly used 2D Inception block. In (b) the whole *InceptionV1* architecture is illustrated. The activation function always is a *ReLU*. Between a convolutional layer and an activation function batch normalization is placed. If not mentioned the stride is 1. Note that average pooling, which is the last layer of the original model, is removed to maintain the spatial structure and get a spatial feature map.

calculates the smoke probability map P_t each frame. Table 5.1 gives an overview of the inputs, outputs and receptive fields for each layer of the *InceptionV1* architecture presented in Figure 5.2(b).

The *InceptionV1* model is famous for its SotA performance on the *ImageNet* dataset. The weights for RGB pretraining on *ImageNet* are publicly available. Since transfer learning shows much better results in many applications, it is applied for RGB input. Detailed design choices of such architectures are typically made by many trials and iterative changes on benchmark tasks like image classification.

Layer	type	output	max receptive field
0	input	$T \times 256 \times 256 \times 3$	–
1	Conv	$T \times 128 \times 128 \times 64$	$1 \times 7 \times 7$
2	Max-Pool	$T \times 64 \times 64 \times 64$	$1 \times 11 \times 11$
3	Conv	$T \times 64 \times 64 \times 64$	$1 \times 11 \times 11$
4	Conv	$T \times 64 \times 64 \times 192$	$1 \times 19 \times 19$
5	Max-Pool	$T \times 32 \times 32 \times 192$	$1 \times 27 \times 27$
6,7	2×2D Inc.	$T \times 32 \times 32 \times 480$	$1 \times 59 \times 59$
8	Max-Pool	$T \times 16 \times 16 \times 480$	$1 \times 75 \times 75$
9-13	5×2D Inc.	$T \times 16 \times 16 \times 832$	$1 \times 235 \times 235$
14	Max-Pool	$T \times 8 \times 8 \times 832$	$1 \times 251 \times 251$
15,16	2×2D Inc.	$T \times 8 \times 8 \times 1024$	$1 \times 379 \times 379$
17	FC (sigmoid)	$T \times 8 \times 8 \times 1$	$1 \times 379 \times 379$

Table 5.1: Complete architecture for single frame approach, feature extractor + classifier. Input is a sequence of length T. The frames are processed distributed over time by the layers, i.e. independently from each other. The FC layer transforms the features to smoke probabilities for each cell and each time step.

The proposed architecture is trained end to end. Since there is no way the architecture could profit from sequences of frames in the training phase in each epoch, one sample is extracted out of each sequence. Each batch consists of 162 sequences with a length of 1. The learning rate at the beginning is $\rho_0 = 0.001$ and decayed each epoch exponentially by $\delta = 0.995$. The training is aborted, when the training loss does not change anymore, which is after 1500 epochs. Every 25th epoch the model is evaluated on the test set.

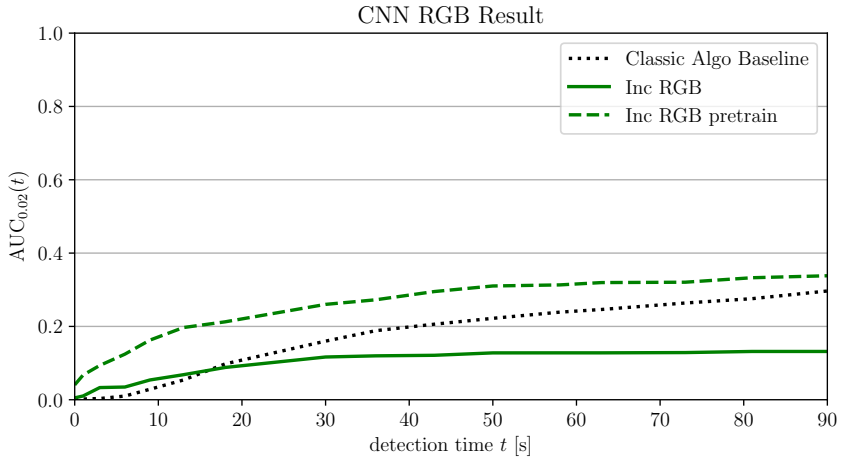
The best result on the test set is illustrated in Figure 5.3.

The dataset is not rich enough to get adequate results on RGB input on it. They are even worse than the baseline. With pretrained weights the baseline using a classic CV algorithm is outperformed by far. There is still space for improvement left, which is tackled in the next section by adding temporal information.

5.2 Temporal Input for CNN

The first idea to use temporal information for DL-VSD methods is to feed Diff or Flow input instead of RGB into a CNN. Figure 5.4 shows examples for these input modalities.

Diff and Flow contain low level temporal information, which are intuitively very characteristic for smoke.



Model	**Input**	$D_{\textbf{Speed}}(0.02)$
Classic Algo	RGB/Flow	0.1814
Inc	RGB	0.1076
Inc pre	RGB	0.2670

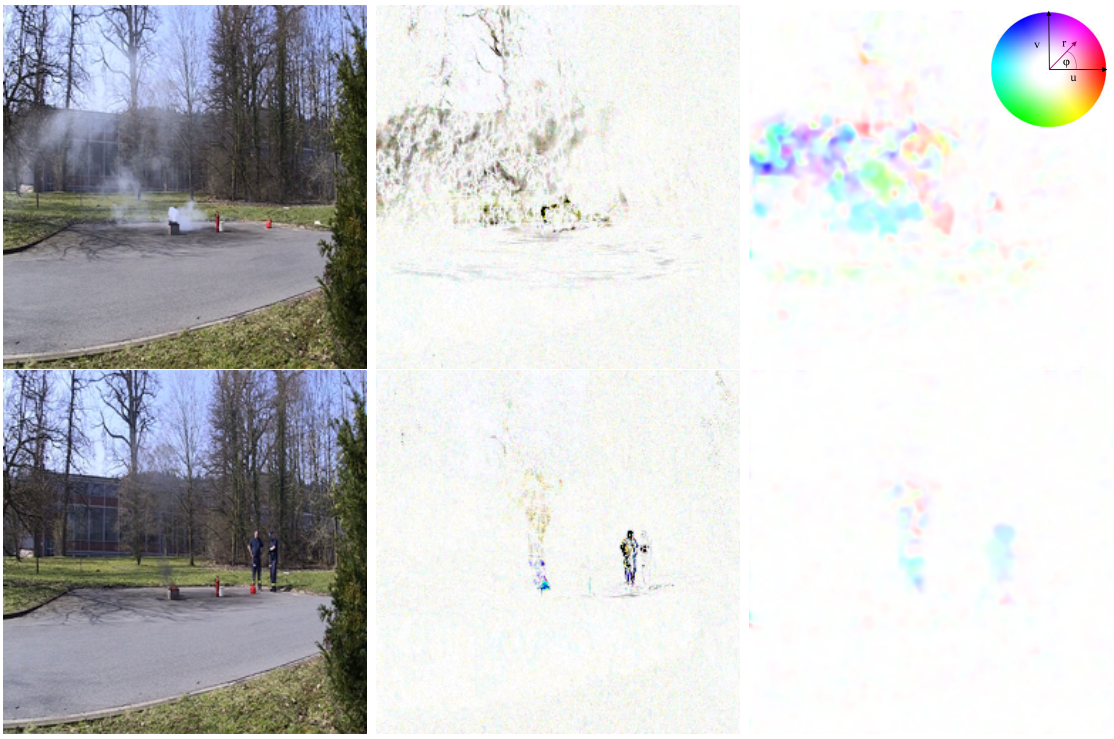
Figure 5.3: The resulting $\mathrm{AUC}_{0.02}(t)$ curves for the *InceptionV1* (Inc) approach using RGB input. For comparison also the classic algorithm is added. The table summarizes the evaluation measure ($D_{\mathrm{Speed}}(0.02)$).

To estimate probabilities for smoke the same CNN architecture as in Section 5.1 is chosen. The backbone CNN is an *IncepetionV1*. It extracts features, which are fed into an FC layer to calculate probabilities of smoke (see 5.1).

Diff and Flow information are extracted for each two consecutive frames in a sequence. Differences are calculated on the fly during training and evaluation, since the computational complexity is very small in comparison to all other computing steps. The resulting range of each channel is $\{-255, \ldots, 255\}$.

Optical Flow is calculated using *TV-L1* [61] from OpenCV with the default parameterization. The computational complexity is very high, so that it is not suitable to calculate it on the fly. Since precalculating the Flow and store it as 32 bit floating-point number is too memory consuming, the resulting (u, v) vectors are clipped to the range $[-10, 10]^2$ and normalized to the range $\{0, \ldots, 255\}^2$. Finally, the Flow is stored with 8 bit for each of the two channels in each pixel. Clipping is done to resolve the velocity of slow-moving objects higher since smoke is slow.

Training is conducted with the same hyperparameters as for RGB input. Neither for Flow nor Diff pretraining is available. The best result on the test set is illustrated in Figure 5.5.



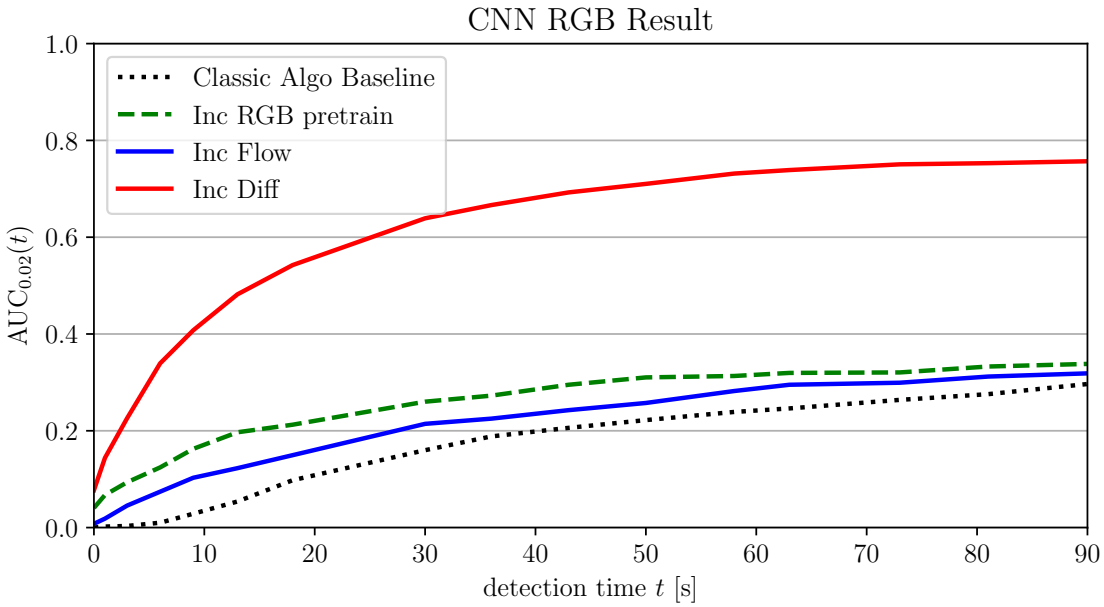
(a) (b) (c)

Figure 5.4: Visualization of different input modalities for the DL-VSD systems. (a) RGB image with resolution 256×256. (b) Diff images calculated by two consecutive 256×256 RGB images with a temporal distance of 333ms. For visualization purpose the absolute value of the resulting difference image is inverted. (c) Flow images calculated by two consecutive 256×256 RGB images with a temporal distance of 333ms. For visualization purpose (u, v) vectors are transformed into polar coordinates (r, φ) and mapped to the HSV color space.

The CNN with Flow input is better than the baseline using a classic CV algorithm, which is also based on Flow. Since both algorithms have similar input information (the classic Algo even more) one can conclude, that the handcrafted model based on physical smoke properties is worse than statistically learned properties.
Nevertheless, RGB with pretraining is better than Flow, which shows the benefit of pretraining. The Diff input is by far the best. Reasons for this result are discussed in Section 5.6.

5.3 Temporal Accumulation of CNN Prediction

Smoke usually has the expanding property, i.e., if smoke is in a cell at time t, it is very likely that it also exists in this cell for the next frames. This assumption helps to reject smoke similar objects, which leave cells after a few frames, like persons



Model	**Input**	$D_{\textbf{Speed}}(0.02)$
Classic Algo	RGB/Flow	0.1814
Inc pre	RGB	0.2670
Inc	Flow	0.2252
Inc	Diff	0.6313

Figure 5.5: The resulting $AUC_{0.02}(t)$ curves for the *InceptionV1* (Inc) approach using Diff and Flow input. For comparison, the classic algorithm and the pretrained single-frame approach with RGB input are added. The table summarizes the evaluation measure ($D_{\text{Speed}}(0.02)$).

with white shirts. The robustness against local stable objects with random dynamic textures, like leaves in the wind or floating water, could also be improved.

Two intuitive ways to take benefits from this property are: If the features are smoke like in frame t, then be less sensitive in frame $t+1$. If the smoke feature is moderate smoke like over a couple of following frames increase the smoke probability from frame to frame.

According to this intuition, this section introduces a straight forward handcrafted concept to extend the single-frame approach: The idea is to accumulate the cell-wise smoke probabilities over time by a recurrent low pass filter

$$P_t = (1-\gamma)P_{t-1} + \gamma\tilde{P}_t, \tag{5.3.1}$$

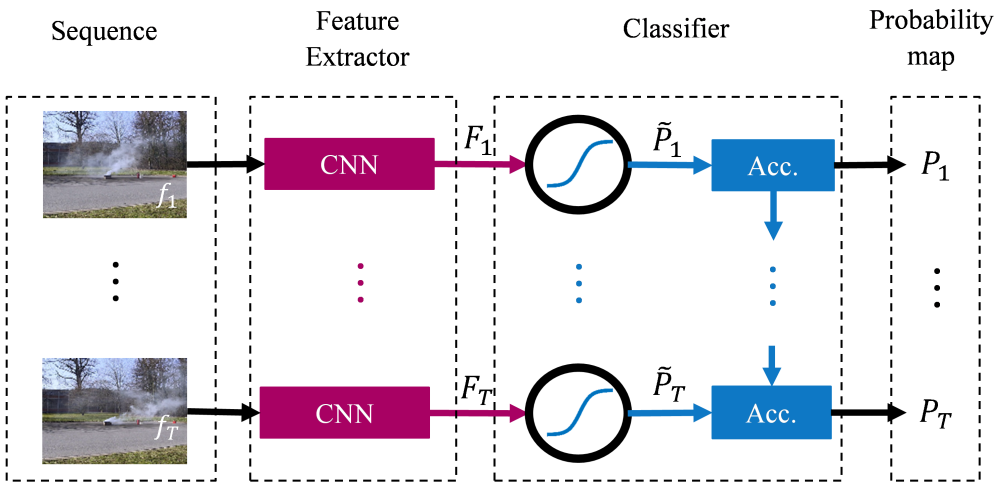
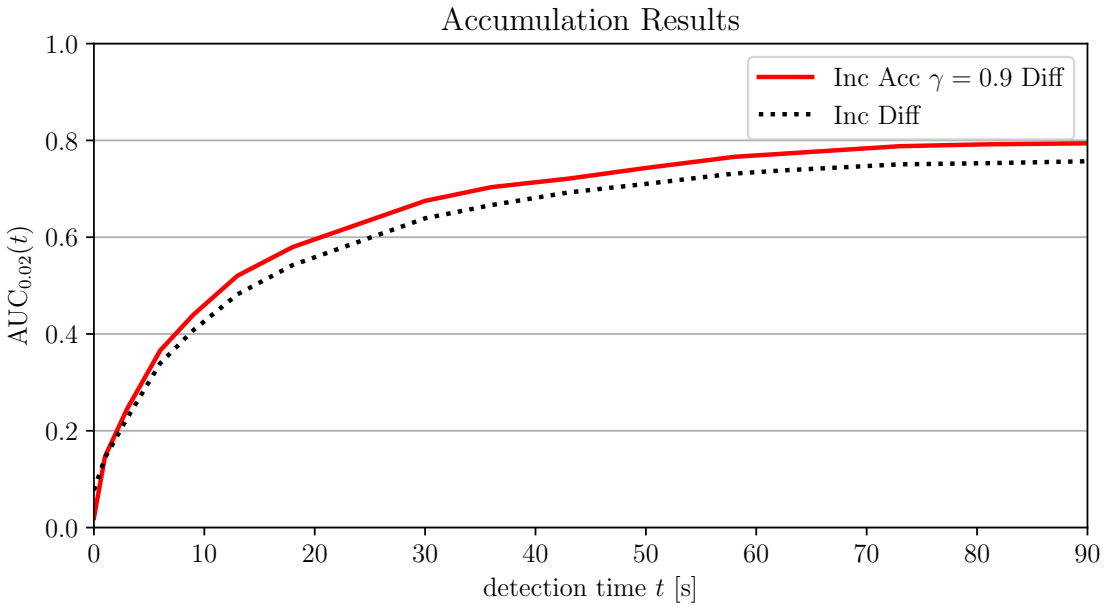


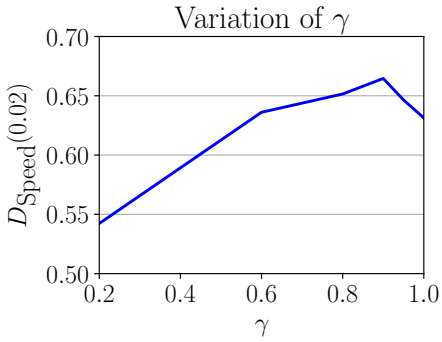
Figure 5.6: The arrows show the information flow of the single frame concept. First spatial image features $F_1, \ldots, F_T$ are extracted of the sequence $f_1, \ldots, f_T$. These feature are used by a sigmoid layer to calculate frame-wise probability maps $\tilde{P}_1, \ldots, \tilde{P}_T$ for smoke. And the frame-wise probabilities are then accumulated (Acc.) over time to get more robust results $P_1, \ldots, P_T$.

where $\tilde{P}_t$ is the single-frame based smoke prediction of frame t and γ is the time constant. Figure 5.6 visualizes the information flow. Note that P_t is not clipped according to a lower bound of $\tilde{P}_t$, which could be a possible extension to filter out some noise. The *InceptionV1* with a FC layer trained in the last section on single frames with Diff input is utilized. The weights of the backbone CNN, *InceptionV1*, are fixed, and only the sigmoid layer is trained. To investigate the influence of the time constant different γ are tested to find the best.

For training, a sequence length of 54 and a batch size of 3 is applied. The learning rate started at $\rho_0 = 0.001$ and is decayed each epoch by a factor of $\delta = 0.8$. The test set is evaluated each epoch. The training runs for 10 epochs. The initial smoke probability map P_0 is set to 0.5 to stabilize the training. The results are illustrated in Figure 5.7.

The additional temporal information increases the performance in *Detection Speed*. Choosing $\gamma < 0.6$ the model becomes much slower such that the $\text{AUC}_{0.02}(t)$ is less than the original model for low detection times t. For $0.6 \leq \gamma < 1.0$ the original model is outperformed and the accumulation increases the *Detection Speed*. And for $\gamma = 0.9$ the best result is obtained. Note that the same procedure for RGB and Flow input also leads to a similar improvement. In Section 5.6 these observation are embedded in a consideration of long and short term dependencies for DL-VSD methods.





Model	**Input**	$D_{\mathbf{Speed}}(0.02)$
Inc pre	RGB	0.2670
Inc Acc pre	RGB	0.2863
Inc	Flow	0.2252
Inc Acc	Flow	0.2412
Inc	Diff	0.6313
Inc Acc	Diff	0.6646

Figure 5.7: The upper graphic shows the resulting $\mathrm{AUC}_{0.02}(t)$ curve for the best accumulation (Acc) model. For comparison also the *InceptionV1* without accumulation is added. The lower left illustration shows the *Detection Speed* with respect to different time constants γ. Note that for $\gamma = 1$ the model is the origin *InceptionV1* network. The table compares the result without accumulation to the one with accumulation for each input modality.

5.4 CNN+LSTM for VSD

The intuition behind a CNN+LSTM architecture is to first extract frame-wise spatial features by a CNN and to use these spatial features to extract temporal information by an LSTM. Figure 5.8 shows the information flow.

The feature extractor consists of two parts: a CNN, which is again an *InceptionV1*

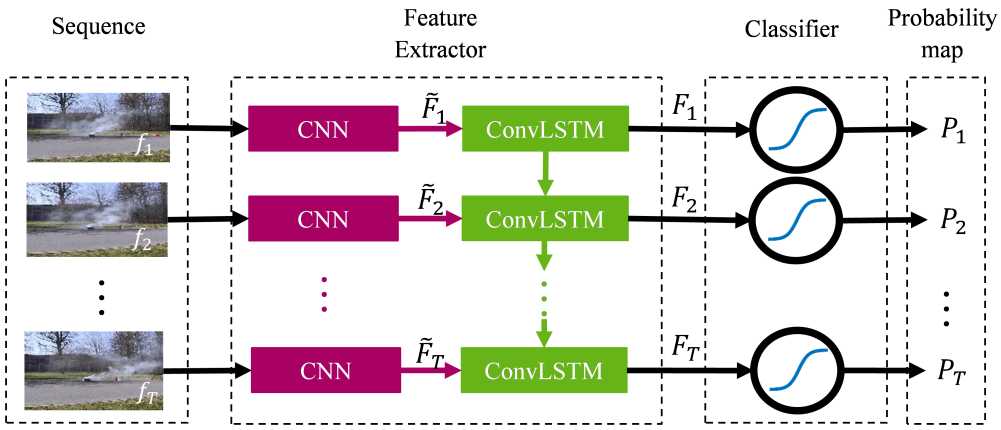


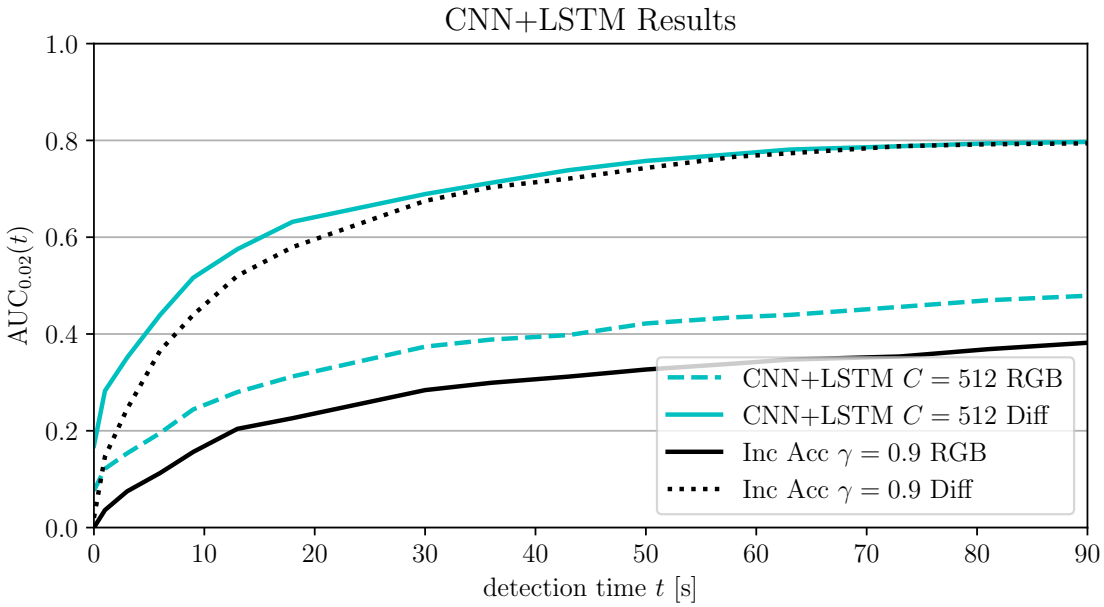
Figure 5.8: The arrows show the information flow of the CNN+LSTM concept. First spatial image features $\tilde{F}_1, \ldots, \tilde{F}_T$ are extracted of the sequence $f_1, \ldots, f_T$. These feature are passed to a 1×1 ConvLSTM, which combines the spatial with temporal information to spatio-temporal features $F_1, \ldots, F_T$. Finally a sigmoid layer is use to calculate the resulting probability maps for each frame $P_1, \ldots, P_T$.

and an LSTM, which works cell wise and is therefore realized by an 1×1 ConvLSTM. As classifier a FC layer with sigmoid activation is appended. Note that this architecture theoretically can use information of all past frames for the current smoke prediction. The 1×1 ConvLSTM extracts c temporal features for each cell in each frame. Different c values are investigated.

It turns out that training this architecture to get better results than using the accumulation approach, is challenging. However, the following iterative strategy finally outperformed the accumulation approach: Start with a sequence length of 3 frames and a batch size of 60, change to 6 frames and a batch size of 30, afterwards 12 frames and a batch size of 15 and finally a sequence length of 24 and a batch size of 7. For each batch configuration train the model for 100 epochs. Start with the learning rate $\rho_0 = 0.001$ and use a decay factor of $\delta = 0.99$. Evaluate the test set every 5 epochs. The best results are illustrated in Figure 5.9.

As CNN the *InceptionV1* is utilized. The results are created for the Diff and RGB input images. For RGB input weights pretrained *ImageNet* are chosen. The first surprising insight is that Diff input performs significantly better than RGB even though pretraining for RGB is available. In theory, the CNN+LSTM should be able to learn the same information, which is extracted from Diff images, since the difference operation is linear. This behavior is analyzed in Section 5.6.

Furthermore, using Diff input the CNN+LSTM outperforms all of the currently



Model	**Input**	$D_{\textbf{Speed}}(0.02)$
Inc Acc pre	RGB	0.2863
CNN+LSTM pre	RGB	0.3775
Inc Acc	Diff	0.6646
CNN+LSTM	Diff	0.6903

Figure 5.9: The graphic shows the resulting $AUC_{0.02}(t)$ curve for the best CNN+LSTM models. $c = 512$ performed best ($c = 32, 64, 128, 256, 1024$ are also tested). For comparison also the *InceptionV1* with accumulation is added. The table summarizes the results.

investigated approaches.

5.5 3D-CNN concept

Up to now, the investigated temporal architectures do not show a significant improvement compared to the single frame architecture with Diff input. The CNN+LSTM and the accumulation approach can use long term information to detect smoke, from which they do not benefit. In this section, a 3D-CNN approach is investigated. Since the temporal receptive field of 3D-CNNs is fixed, it completely concentrates on short term information.

For investigations the *i3D* is chosen, which achieves SotA results on action recognition tasks. The temporal receptive field of *i3D* is limited to a fixed length, i.e. R frames. For each frame except the first and the last $R/2$ of a sequence a feature

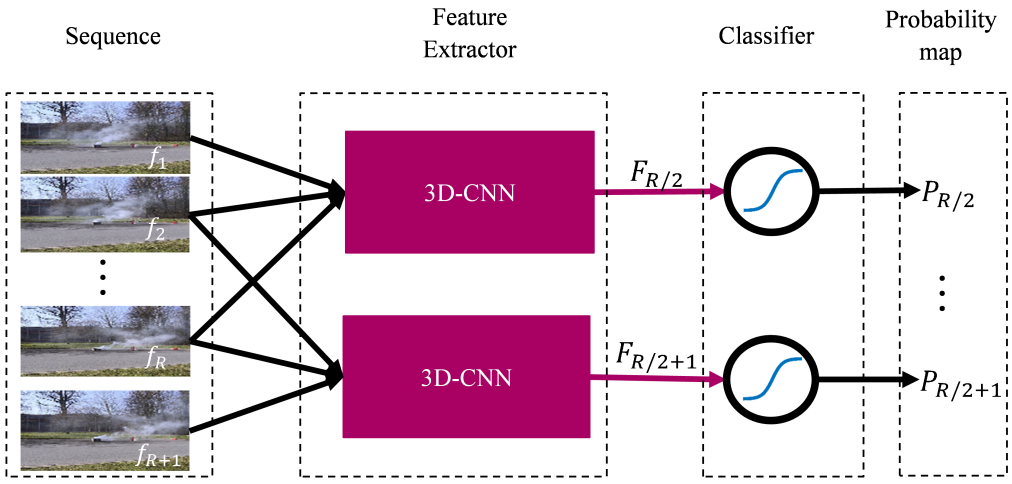
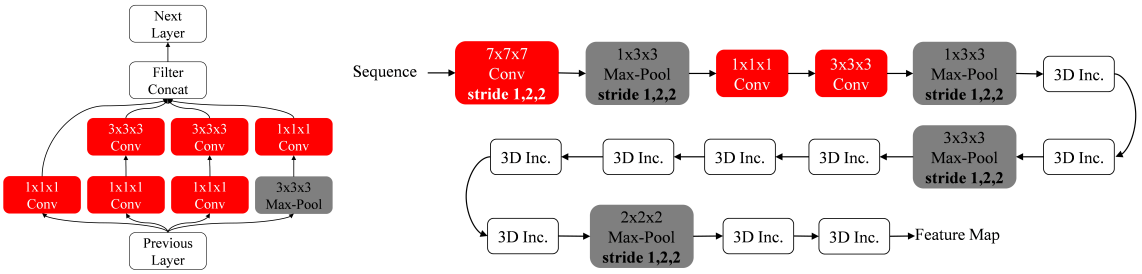


Figure 5.10: The arrows show the information flow of the 3D-CNN concept. A fixed sequence of length R (=temporal receptive field of 3D-CNN architecture) is propagated through the 3D-CNN network, which extracts spatio-temporal features $F_{R/2}$. These features are passed to a sigmoid layer, which calculates the resulting probability map $P_{R/2}$ for the $R/2$-th frame.

map is calculated. From these feature maps smoke probability maps are calculated by a FC layer. Figure 5.10 shows the information flow.

The *i3D* is constructed by inflating the CNN *InceptionV1* architecture into a 3D version. This is done by replacing 2D convolutions through 3D convolutions and 2D pooling by 3D pooling. The resulting inflated Inception module is illustrated in Figure 5.11.



(a) (b)

Figure 5.11: The *i3D*: (a) the repeatedly used 3D Inception block and (b) the architecture. The activation functions are always a *ReLU*. Between convolutional layer and activation functions batch normalization is placed. The stride is of the form time,x,y. If not mentioned, the stride is 1. The *i3D* architecture is similar to the *InceptionV1* (see Figure 5.2), but all 2D convolutions and poolings are inflated to 3D versions.

Normally, the receptive field of the *i3D* is $r = 99$ frames (33s), since some of the 3D layers of the original *i3D* architecture have also in the temporal dimension a

stride of 2. For the investigations of this thesis the original architecture is slightly modified and the strides in the temporal dimension of all layer are set to 1 (Figure 5.11). This modification reduces the temporal receptive field to $r = 30$ frames (10s). Additionally the padding is modified, such that each frame yields a feature map, but for the first and the last $r/2$ frames there is not the full receptive field of informations. The resulting architecture is shown in Table 5.2. Since the *i3D*

Layer	**type**	**output**	**max receptive field**
0	input	$T \times 256 \times 256 \times 3$	–
1	3D Conv	$T \times 128 \times 128 \times 64$	$7 \times 7 \times 7$
2	3D Max-Pool	$T \times 64 \times 64 \times 64$	$7 \times 11 \times 11$
3	3D Conv	$T \times 64 \times 64 \times 64$	$7 \times 11 \times 11$
4	3D Conv	$T \times 64 \times 64 \times 192$	$9 \times 19 \times 19$
5	3D Max-Pool	$T \times 32 \times 32 \times 192$	$9 \times 27 \times 27$
6,7	2×3D Inc.	$T \times 32 \times 32 \times 480$	$13 \times 59 \times 59$
8	3D Max-Pool	$T \times 16 \times 16 \times 480$	$15 \times 75 \times 75$
9-13	5×3D Inc.	$T \times 16 \times 16 \times 832$	$25 \times 235 \times 235$
14	3D Max-Pool	$T \times 8 \times 8 \times 832$	$26 \times 251 \times 251$
15,16	2×3D Inc.	$T \times 8 \times 8 \times 1024$	$30 \times 379 \times 379$
17	FC (sigmoid)	$T \times 8 \times 8 \times 1$	$30 \times 379 \times 379$

Table 5.2: The *i3D* architecture+FC classifier. There is a slight difference to the original *i3D* architecture: Firstly, the temporal stride is always set to 1 to reduce the temporal receptive field from 99 to 30 frames. Secondly, to assure that each frame gets a grid of smoke probabilities the padding is changed.

architecture takes a look into the future by $r/2$ for each frame's decision, a penalty for the calculation of *Detection Speed* is introduced

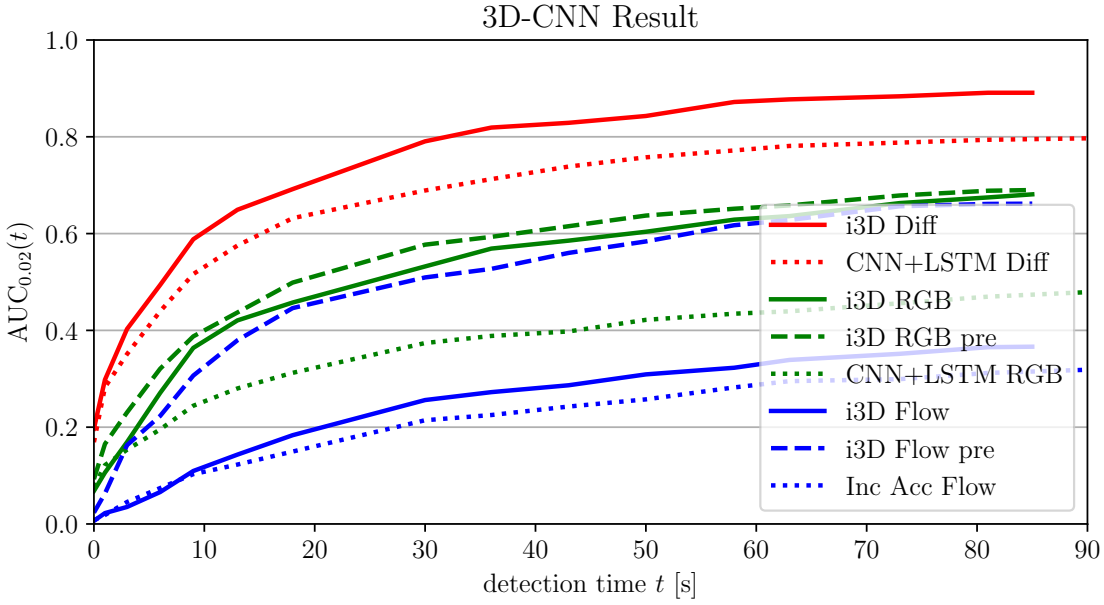
$$D_{\text{Speed}}(\theta) = \frac{1}{270 - r/2} \int_0^{270-r/2} \text{AUC}_\theta(t)\text{d}t, \tag{5.5.2}$$

i.e. the maximal detection time is set to $T = 270 - r/2$ frames. Since $\text{AUC}_\theta(t)$ is an increasing function, this modification decreases the *Detection Speed* and allows a fair comparison to approaches, which only have past information.

All input modalities RGB, Flow and Diff are investigated. For RGB and Flow weights pretrained on *Kinetics* are available.

The sequence length for training is set to 30, since this is the temporal receptive field. Note that each sample has only one frame, which takes benefit of the full information from all other frames. One batch consists of 6 events. The learning

rate is decaying starting at $\rho_0 = 0.001$ each epoch by a factor of $\delta = 0.995$. After each 10th epoch the model is evaluated on the test. The training converges after about 500 epochs. In Figure 5.12 the result is illustrated. Using RGB input, the



Model	**Input**	$D_{\textbf{Speed}}(0.02)$
CNN+LSTM pre	RGB	0.3775
i3D	RGB	0.5357
i3D pre	RGB	0.5650
Inc Acc	Flow	0.2412
i3D	Flow	0.2751
i3D pre	Flow	0.5139
CNN+LSTM	Diff	0.6903
i3D	Diff	0.7696

Figure 5.12: The graphic shows the resulting $\text{AUC}_{0.02}(t)$ curve for the best *i3D* models. For comparison the previously best achieved results are given for each input modality. The table summarizes the results.

result is significantly worse than the one using Diff input. Even though differences are learnable by the first 3D convolutional layer. Similar to the CNN+LSTM this is contradictory. Also, the Flow result is worse than Diff as observed using the single frame CNN approach. These issues are discussed in Section 5.6.

Nevertheless, the *i3D* outperforms all other approaches significantly in all input modalities. For RGB and Flow pretraining on *Kinetics* also improves the results. It is also worth to notice that RGB input is better than Flow but still worse than the currently best result on Diff. *i3D* using difference images is by far the best DL-VSD

model in this thesis.

5.6 Discussion of Temporal DL Results

The analysis of DL-VSD approaches shows that combining spatial with temporal information significantly increases the performance, which is intuitively plausible. However, some observations are surprising: Why do the methods, which can learn, how to deal with temporal information (DL-VSD and 3D-CNN) work better with Diff input than with RGB? Why does the Diff input perform better than Flow? What is more relevant for VSD long or short term information? These questions are discussed now.

5.6.1 Advantages of Temporal Input over RGB

The claim for VSD is that the camera is fixed and there is no movement and not more than slight shaking. Under this circumstances calculating Diff or Flow removes the complete background. Especially spatial structures are suppressed. Only temporal noise is left. Therefore the DL network is directly forced to concentrate on movement. Since smoke is a moving object, even slow or ambient smoke slightly differs from temporal noise, the input of the network contains enough information to detect smoke. However, the challenge is much easier: Distinct smoke only from moving objects.

Furthermore, there is no relevant information for smoke lost.

When only using a CNN it is therefore not surprising that temporal input performs better than RGB. In contrast temporal architectures 3D-CNN and CNN+LSTM theoretically can learn Diff of two images. Why does Diff also work better for them? When considering the training runs of temporal DL architectures with RGB input (Figure 5.13), the training loss converges to a similar value, while the test performance is worse. This behavior is an indicator that the network suffers from overfitting, which could be the reason why also temporal DL-VSD approaches deliver better results using temporal input.

There are more structural details available when using RGB images. These structural details could lead the networks to concentrate on contextual information, which

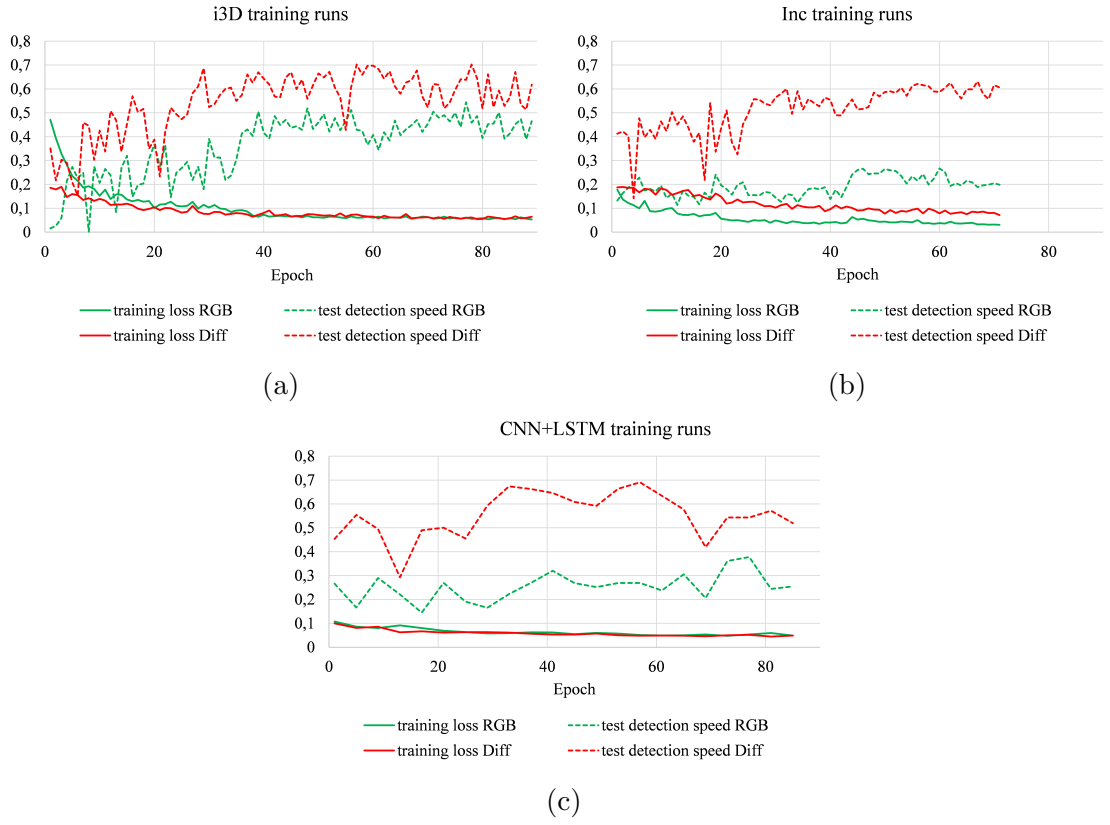


Figure 5.13: Comparison of training runs for RGB and Diff of different network architectures: (a) *i3D*. (b) *InceptionV1*. (c) CNN+LSTM. Note that only epochs are illustrated, in which the test set is evaluated.

has nothing to do with smoke, e.g., tiles in a fire lab. Using temporal input these details are suppressed.

5.6.2 Diff versus Flow input

For humans, information of Flow is much easier to interpret than information of Diff. But the DL approaches reach much better results with Diff input for all tested DL-VSD approaches. One reason could be that Diff images have suitable benefits for VSD. However, one should be careful inferring general conclusions, that Diff is better for VSD than Flow, since the calculation of Flow has many degrees of freedom.

In comparison to Flow images, Diff images contain edge, color and contrast information. Diff images emphasize outer and inner smoke frontiers, changes in color, and the chaotic temporal behavior of contrast, i.e., there are no main spatial frequencies.

These are all very characteristic properties for visual smoke behavior, from which a DL-VSD algorithm with Diff input profits.
The *TV-L1* algorithm for Flow calculation depends on many parameters, which are chosen as the default OpenCV parameters. A structured analysis of these parameters is necessary.
There are other SotA algorithm to calculate Flow, which could be more suitable ones for DL-VSD approaches. An qualitative analysis of different Flow algorithms for VSD with different parameter combinations is done by [54].
The parameterization of the Flow algorithm also depends on the temporal and spatial resolution, which should also be varied for a structured analysis. Furthermore, a clipping and quantization of the resulting (u, v) vectors is conducted, which can remove relevant information. These issues require a comprehensive, structured analysis to get a robust conclusion about Flow as input for a DL-VSD system, but to the author's suggestion, the performance gap between Diff and Flow is too big to be closed by optimization of the Flow hyperparameters.

5.6.3 Short versus Long Term Information

In all temporal approaches, it turns out that short term information is very characteristical for smoke. The success of Diff input in Section 5.2 shows, that one yields high-quality smoke predictions, only using two consecutive frames within 333ms.
The accumulation approach in Section 5.3 balances short and long term information. $\gamma \in [0, 1]$ weights the impact of short towards long term information, i.e., the smaller γ, the higher is the impact of long term information. The conducted experiments show that the result only improves for high values of γ. The best *Detection Speed* is reached for $\gamma = 0.9$, which improved the result without accumulation by 5%. For $\gamma = 0.9$, the impact of an observed frame vanishes after 2 further frames to $< 1\%$.
In the accumulation approach, the long term information is added by a handcrafted method, and there is a fixed weighting introduced, i.e., in every situation long and short term information have not variable impact.
In contrast, the CNN+LSTM approach theoretically can learn, which information is more relevant, dependent on the situation. For example, it should learn to detect smoke with high density due to short term information as well as thin smoke by

using long term information like background noise. Indeed the accumulation result is improved by 5%. But the $AUC_{0.02}(t)$ in Figure 5.9 shows that, the CNN+LSTM approach only performs better for small detection times ($t <= 70$s) and for a detection time $t > 70$s the performance is equal to the accumulation approach, which indicates that the CNN+LSTM deals better with short, but equally with long term information compared to the accumulation method. Considering that the accumulation approach does nearly not use any long term information, one can conclude that the CNN+LSTM approach also does not significantly benefit from it. Another suggestion is that the training sequences are too short (≤ 8s). However, using longer sequences for training leads to worse results.

The *i3D* outperforms every other approach and improved the CNN+LSTM by 11% using the information of 10s for each decision. In the scope of VSD, this is rather short term information, e.g., background noise or slow illumination changes are not considered for an alarm decision. The author assumes that each frame of the 10s period is treated equally for an alarm decision using 3D convolutions, whereas the low pass filter effects in the accumulation approach and the LSTM (see Equation 2.2.7) force the impact of past frames to vanish very fast.

These considerations lead to the question, how can a CNN+LSTM approach be designed and trained to take more benefit of long term information and can this outperform the *i3D*? The investigation of this question requires a deeper insight, of what the network has learned and how the gradient flow impacts recurrent parameters, which is out of the thesis' scope.

5.7 Evaluation on Public Dataset

The analysis of DL-VSD algorithms is done on an internal dataset, which is not publicly available. Therefore it is difficult to rank the proposed approaches to the literature in VSD. The goal of this section is to evaluate the DL-VSD algorithms proposed in this chapter on a suitable public dataset and compare them to other approaches in the literature.

Public datasets for VSD are rare, especially to compare DL approaches, which require a split into training and test set. To the author only the ones of Xu [30]

and *Filonenko* [29] are known. The dataset of Xu consists of single-frame images only. *Filonenko's* dataset is a collection of crops of smoke and negative sequences. For the thesis' approaches, only datasets are relevant, which consists of sequences. Therefore only the *Filonenko* dataset is suitable.

This section starts with a description of *Filonenko's* dataset and the results he obtains with his approach. Afterward, all approaches of this thesis are evaluated and compared on this dataset using *Filonenko's* measures. It can be shown that they all outperform the one of *Filonenko*. Finally, the measures proposed in Section 3.4 are applied, and it turns out that the results differ from the conclusions made on the *Bosch* dataset.

5.7.1 Filonenko's dataset

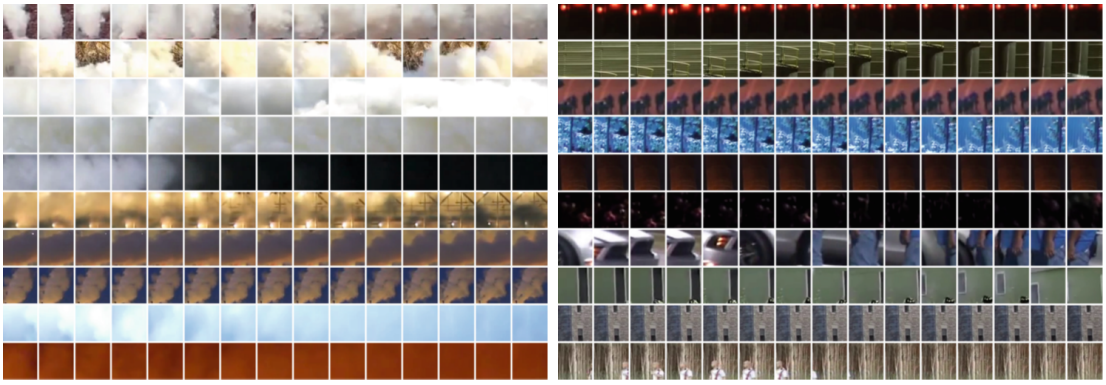


Figure 5.14: Examples for the *Filonenko* sequences. Left shows smoke and right negative sequences.

Filonenko's dataset consists of 162 smoke and 234 negative sequences. These sequences are 64×64 px^2 crops of smoke and non smoke recordings. Some examples are illustrated in Figure 5.14. The rate to which the frames are sampled is unknown, but the author estimates that it is between 15 and 30 fps. Each sequence contains between 65 and 512 frames. Note that training and testing a DL-VSD algorithm on the dataset of *Filonenko* differs from applying it in a real VSD surveillance scenario. Because *Filonenko* uses crops, i.e. small parts of an image, for training and testing. In applications one has to further extend the algorithm and combine the predictions of all crops to a smoke alarm decsion for the whole image.

5.7.2 Filonenko's Measures and DL Approach

Filonenko randomly divided the dataset into a training and test set using a ratio of 80 to 20. It is not clear, which sequence belongs to which set. He investigated a conceptually similar architecture compared to the proposed CNN+LSTM approach. Firstly a custom-designed CNN extracts a feature map for each frame and secondly this feature map is fed into a gated rectified unit (GRU), which is a specific RNN having good gradient behavior like LSTM, but less parameters. In contrast to the proposed approaches in this thesis, the network only predicts the last frame of a sequence during training.

For evaluation on the test set only the first fixed number of frames at the beginning of each sequence is considered. This fixed number of frames is varied from 2 to 67, and each time the accuracy is calculated. Finally, *Filonenko* chooses the maximal

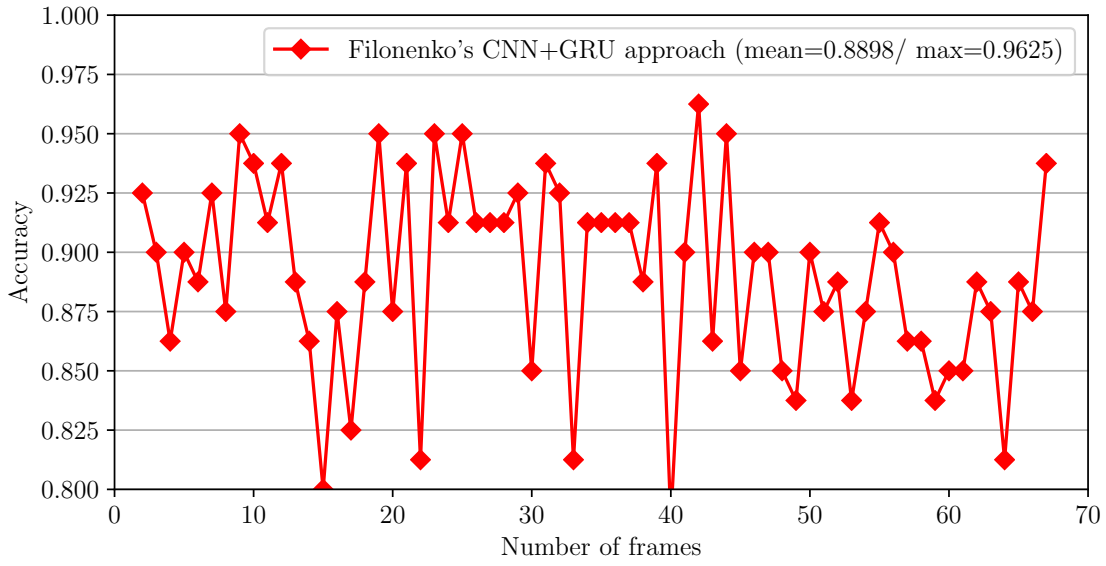


Figure 5.15: Result of *Filonenko's* approach on his test set, which is randomly chosen from his complete data. He applied a CNN-GRU approach and analyzed the accuracy, which depends on the number of frames from the beginning of the sequence. The diamond boxes are estimated from *Filonenko's* paper.

accuracy, which is 96.25% after 42 frames, as the measure for comparison. The accuracy is defined as the ratio of correctly classified to all sequences of crops.

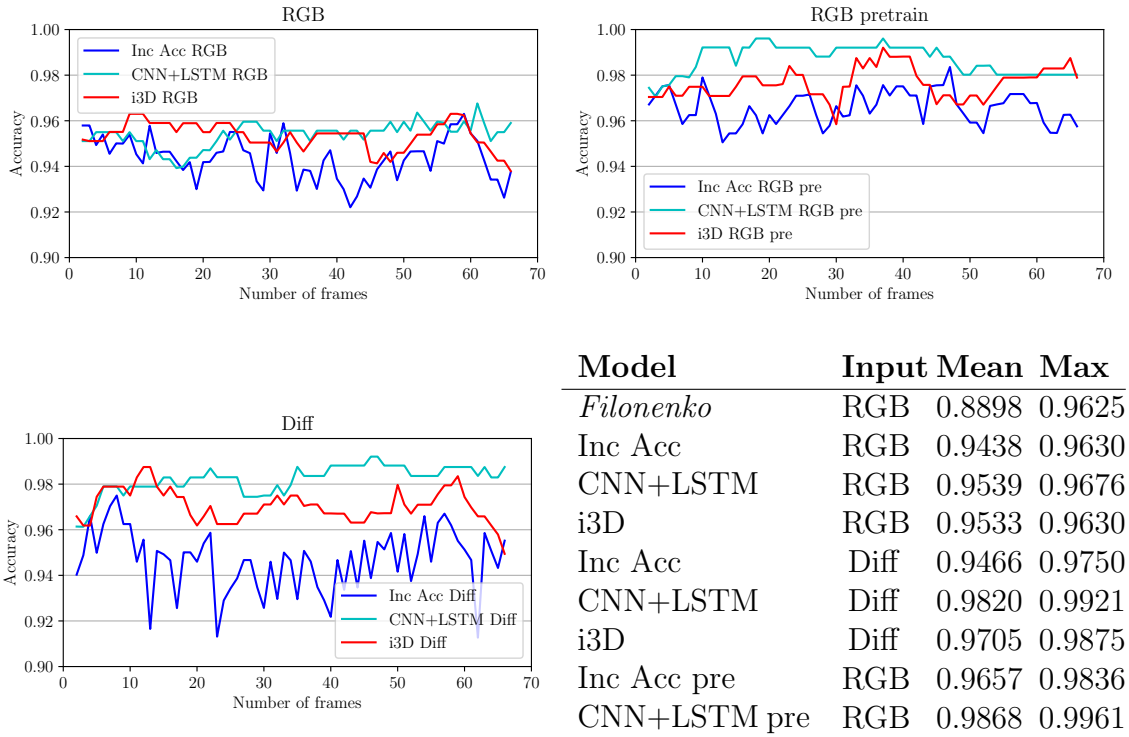
The weakness of this measure is that each data point is determined on a completely different test set since only the first part of the sequence is chosen. Increasing the sequence length should lead to a higher reliability in smoke detection, but since the

negatives are also cropped, critical parts, which may begin later, could be truncated. Therefore Figure 5.15 seems contradictory because the accuracy has a high variation and is not increasing with the number of frames used.

A more comparable measure would be the mean over these accuracies, which is 88.98%.

5.7.3 Comparison of Proposed DL Approaches to Filonenko's

Now the investigated approaches are applied to the dataset of *Filonenko*, i.e., they are completely new trained and tested using the measure of *Filonenko*. Three random splits are generated and evaluated independently, to overcome the issue that the exact split in training and test set done by *Filonenko* is not clear. Finally, the mean over the three test sets is used for comparison. In Figure 5.16 results for



Model	Input	Mean	Max
Filonenko	RGB	0.8898	0.9625
Inc Acc	RGB	0.9438	0.9630
CNN+LSTM	RGB	0.9539	0.9676
i3D	RGB	0.9533	0.9630
Inc Acc	Diff	0.9466	0.9750
CNN+LSTM	Diff	0.9820	0.9921
i3D	Diff	0.9705	0.9875
Inc Acc pre	RGB	0.9657	0.9836
CNN+LSTM pre	RGB	0.9868	0.9961
i3D pre	RGB	0.9765	0.9921

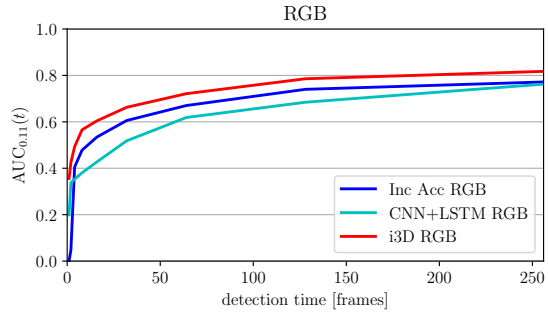
Figure 5.16: Evaluation of all temporal architectures proposed in this thesis on the dataset of *Filonenko* using his measure. Diff and RGB input are tried. The table summarizes the results.

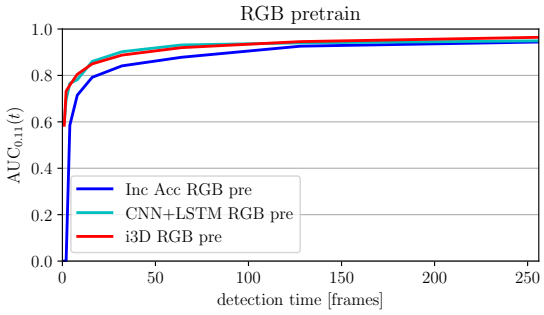
all investigated DL approaches using temporal approaches are illustrated. Diff and RGB input are used. For RGB input, pretraining is also investigated. Without

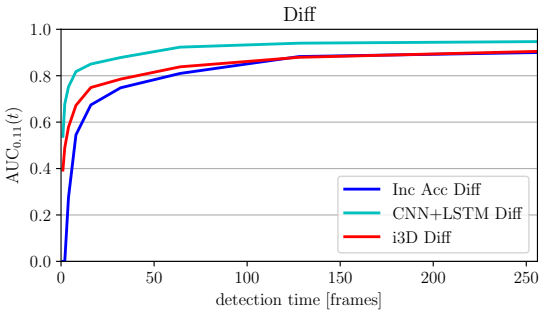
pretraining, the Diff input performs best. However, if one uses pretraining, the results significantly improve, which is not surprising, since the smaller the dataset, the higher the benefit of pretraining.
In contrast to the results on the *Bosch* dataset now the CNN+LSTM approach achieves the best performance for all input modalities followed by the *i3D*.
The achieved result is also better than the one of *Filonenko*. Furthermore, the curvatures have significantly less variation. Nevertheless, one has to be careful by comparing the results, because *Filonenko's* exact split into training and test set is unknown and statistically approximated by the three randomly chosen splits.

5.7.4 Evaluation of Thesis' Measures on Filonenko's Dataset

Now the *Detection Speed* measure from the thesis is applied to the dataset of *Filonenko*. As maximal valid detection time, 256 frames are chosen, and the ROCs are restricted to a maximal FPR of 0.11, which covers the 5 most critical sequences of the test set. For this investigation also the three random splits of the last section are investigated, and for each sensitivity parameter λ the mean over the three FPR and TPR is calculated to construct the ROC. It is not possible to evaluate the thesis' measure on the *Filonenko's* approach, since maximal scores for each frame and of each sequence are necessary, which are not available. The result is illustrated in Figure 5.17. When considering the input modalities, the result is the same compared to the outcome using *Filonenko's* measure: RGB with pretraining is the best followed by Diff and RGB without pretraining is worst.
Considering the DL approach the result changes: Now the *i3D* achieves the best results on RGB without pretraining. The CNN+LSTM stays the best for Diff input and for RGB with pretraining the results of both models are almost identical. Again it is no surprise that RGB with pretraining delivers the best results since the dataset is small. The author's assumption that the CNN+LSTM approach works better than the *i3D* is that all frames in one sequence have the same label. So there is an easy rule to profit from long term information.







Model	**Input**	$D_{\textbf{Speed}}(0.11)$
Inc Acc	RGB	0.6943
CNN+LSTM	RGB	0.6477
i3D	RGB	0.7476
Inc Acc	Diff	0.8255
CNN+LSTM	Diff	0.9187
i3D	Diff	0.8489
Inc Acc pre	RGB	0.8875
CNN+LSTM pre	RGB	0.9231
i3D pre	RGB	0.9256

Figure 5.17: Evaluation of all temporal architectures proposed in this thesis on the dataset of *Filonenko* using the *Detection Speed* measure $D_{\text{Speed}}(0.11)$. The table summarizes the results.

5.8 Qualitative Analysis

Several DL-VSD methods are trained and evaluated on the test set using a quantitative measure, the *Detection Speed*. Now the question is, in which situations smoke is not detected, or the algorithms are sensitive to false alarms.

These critical smoke and negative events are analyzed due to their environmental and recording conditions. It is not surprising that it turns out that events containing only very thin and barely moving smoke and environments with bad or rapidly changing illumination are challenging. Critical negatives usually are events with very slow expanding objects, shadows, or reflections.

5.8.1 Analysis of Critical Smoke Events

The restricted ROC of all temporal models using Diff input is considered in Figure 5.18 to define the critical smoke events.

The *i3D* shows best performance and has the ability to detect 87% of the smoke events without false alarm within 90 seconds, the maximum required detection time.

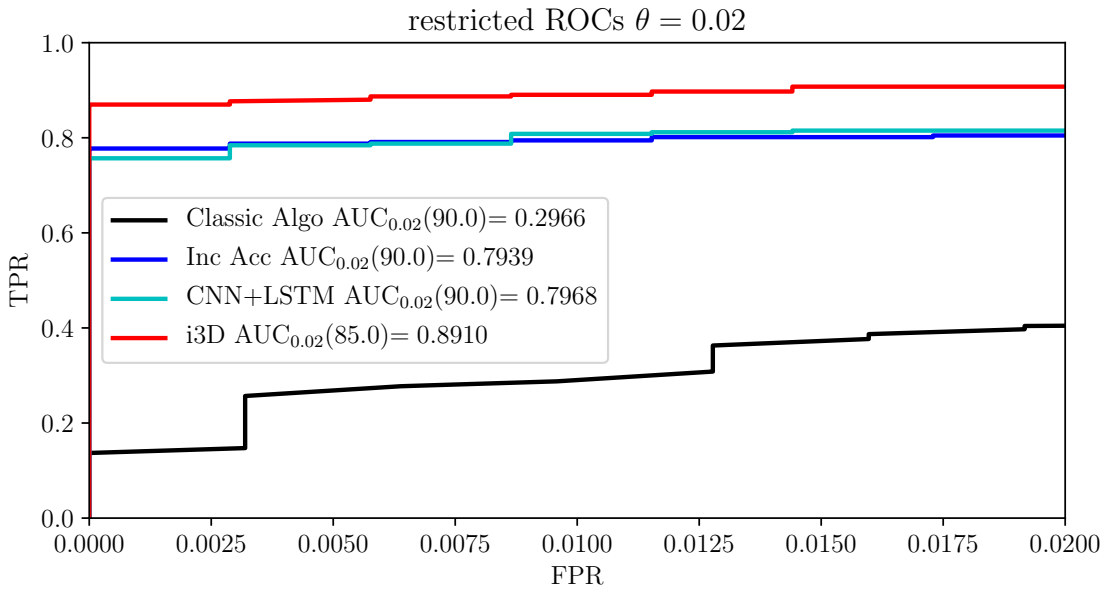


Figure 5.18: Restricted ROCs of all temporal approaches with Diff input. The maximal allowed detection time of 90 seconds is used. Note that the *i3D* determines the smoke probabilities by knowing the next 5 seconds. Therefore the detection time for the *i3D* is reduced to 90 seconds (see also Equation 5.5.2).

The other 13% of smoke events are defined as critical. The distribution of smoke types and environmental conditions within the critical and not critical events are compared due to meaningful aspects in Figure 5.19.

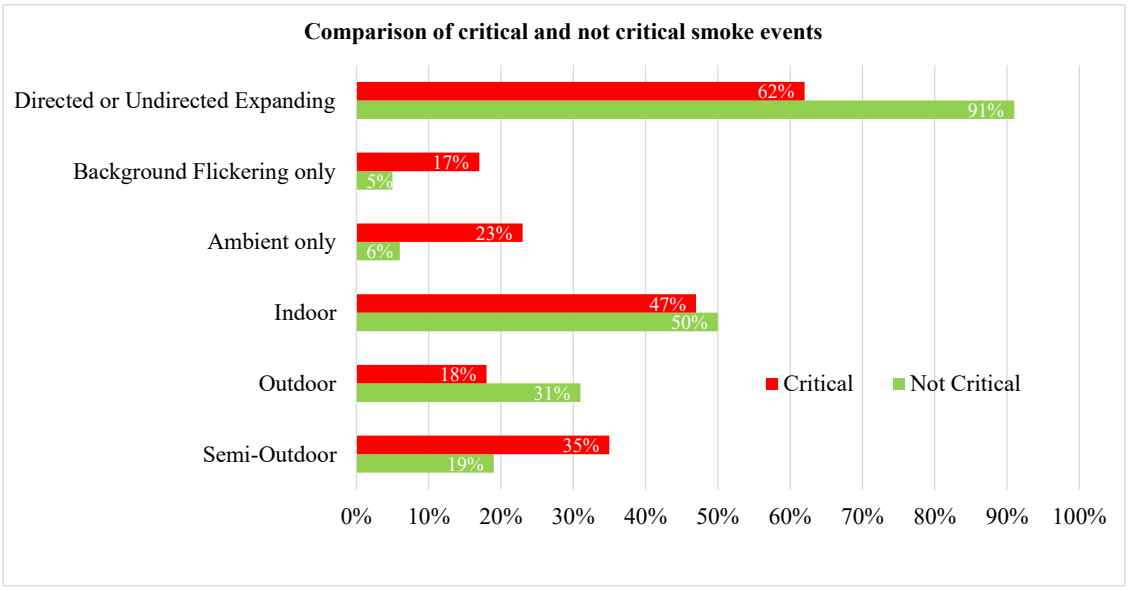


Figure 5.19: Comparison of meaningful aspects in distribution of smoke types and recording conditions. The percentage values are given in relation to all events in each column, e.g. 91% of the not critical smoke events and 62% of the critical smoke events have the DE or UE property.

There are significant differences in smoke types. In almost all not critical events DE

or UE smoke is recognizable. Whereas in 38% of the critical events only have the BGF and/or AM property. This is an indicator that the *i3D* has difficulties with very thin, not moving smoke. Figure 5.20 shows two examples.

(a) (b)

Figure 5.20: Examples for critical smoke events. (a) Very thin BGF smoke in upper right corner. (b) Not moving AM smoke.

Since Figure 5.20a is even hard to detect for humans, it is not a surprise that it is also challenging for the *i3D*. Whereas figure 5.20b shows for humans easy detectable smoke, but for *i3D* it is not detectable, because Diff input is used and in the rare cases, where smoke does not have inner movement, there is not the characteristic change from one frame to the other. Figure 5.19 also shows, that beyond the smoke behavior also the environmental conditions play an important role. The ratio of indoor events is similar, which indicates that in closed rooms or halls with constant illumination, the smoke type is more relevant.

There are significantly fewer outdoor events within the critical smoke recordings. The outdoor events usually have a strong and stable illumination and low dynamic range, since the recordings are taken during daylight, which leads to less temporal noise. Since the smoke movement normally is in the magnitude of noise, the detection is much easier.

Semi-outdoor events are overrepresented in the critical smoke sequences. Critical semi-outdoor environments are rather dark areas, like tunnels or train stations, where blinking or moving lights, headlights of cars and trains, cause global light changes, which suppress the changes caused by smoke (examples in Figure 5.21). This light changes are especially a problem for algorithms using several seconds for an alarm decision like the *i3D*.

Despite these critical smoke events, the algorithm shows impressive performance. It

(a) (b)

Figure 5.21: Typical global light changes in semi-outdoor scenarios, which lead to challenging smoke detection. (a)Blinking lights during smoke event in a tunnel. (b) Headlights of an arriving train behind smoke.

often detects smoke events faster than humans and or in situations a human has to look twice.

5.8.2 Analysis of Critical Negative Events

Critical negative sequences are defined considering the not restricted ROCs in Figure 5.22. The critical negative events are defined according to the ROC of the *i3D*. From the ROC curve of the *i3D* see that the FPR is 26% for TPR $>$ 99%. This means, that there are 26% negative events, which cause a false alarm before nearly all smoke events are detected. These events are defined as critical negatives. Figure 5.23 shows relevant distributions of negative types and environmental conditions within the critical and not critical negative events. Especially events similar to DE and UE smoke are critical for the DL algorithm. When investigating some examples, one can recognize that these are slowly expanding objects with a chaotic texture, e.g., reflections of a rising crane or shadows of opening doors. Blinking lights are also critical, especially yellow ones because in the training set are some flames, which are surrounded by smoke and therefore are labeled as smoke. Negative events with flickering behavior, like trees in the wind or a lake with rough water and static objects, which are visually similar to smoke like clouds, do not cause false alarms. However, some complicated weather conditions like snowing are critical.

The deviations in environmental conditions seem to be transitive correlated to the negative types, because DE or UE events, like cranes or elevators, are usually recorded indoor.

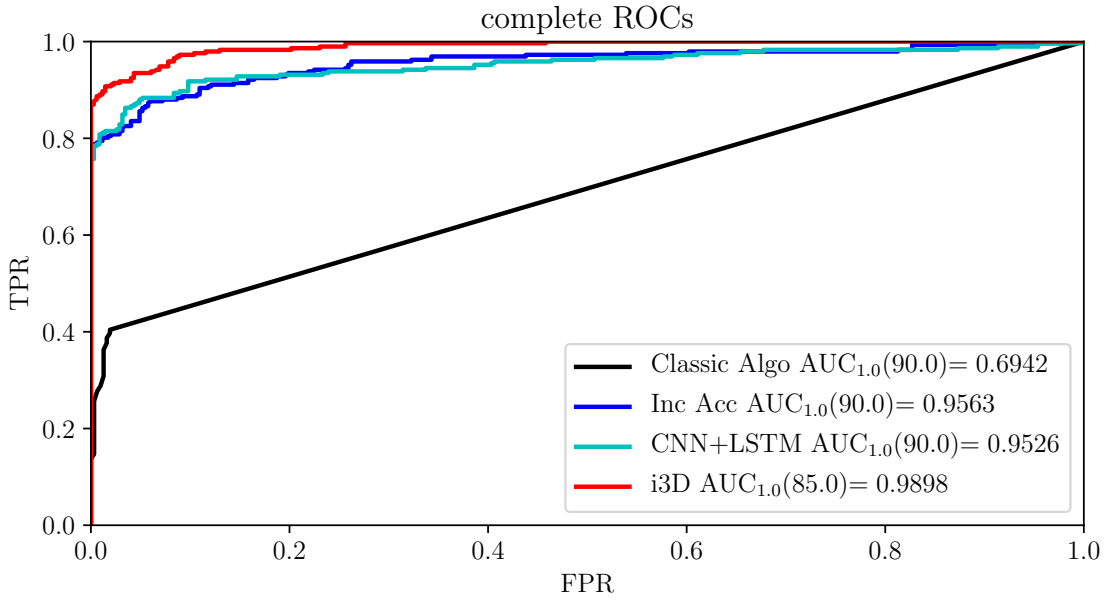


Figure 5.22: Not restricted ROCs of all temporal approaches with Diff input. The maximal allowed detection time of 90 seconds is used. Except for the *i3D*, which uses 85s. Note that the configurable sensitivities of the Classic Algo allow a calculation of a few points on the ROC only. The remaining points are interpolated. This results in the dominant linear slope within the Classic Algo curve.

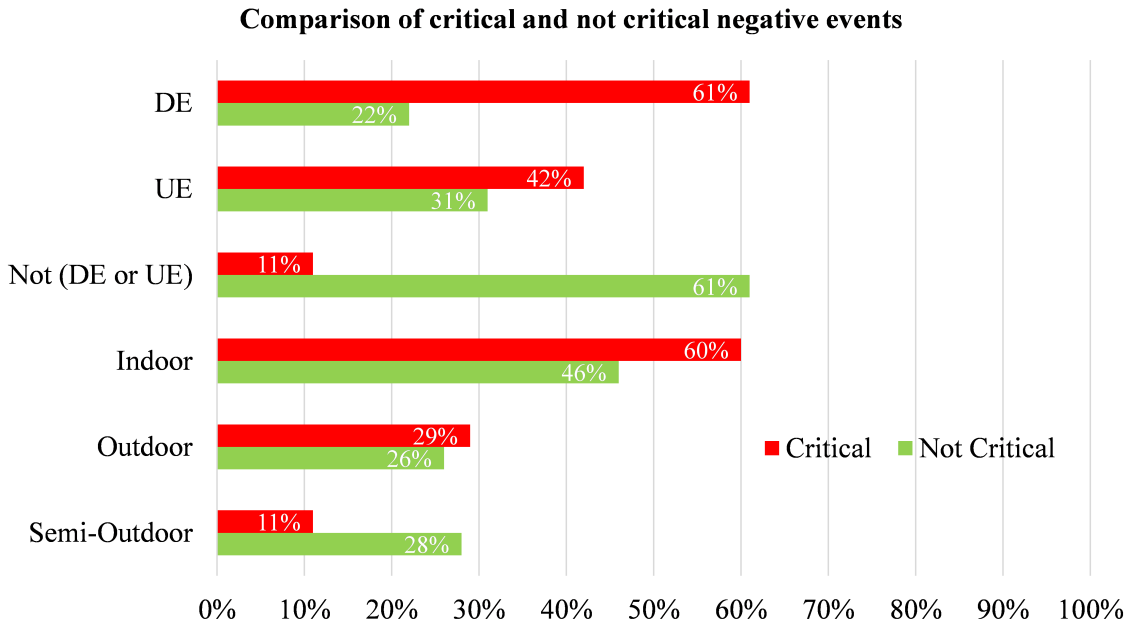
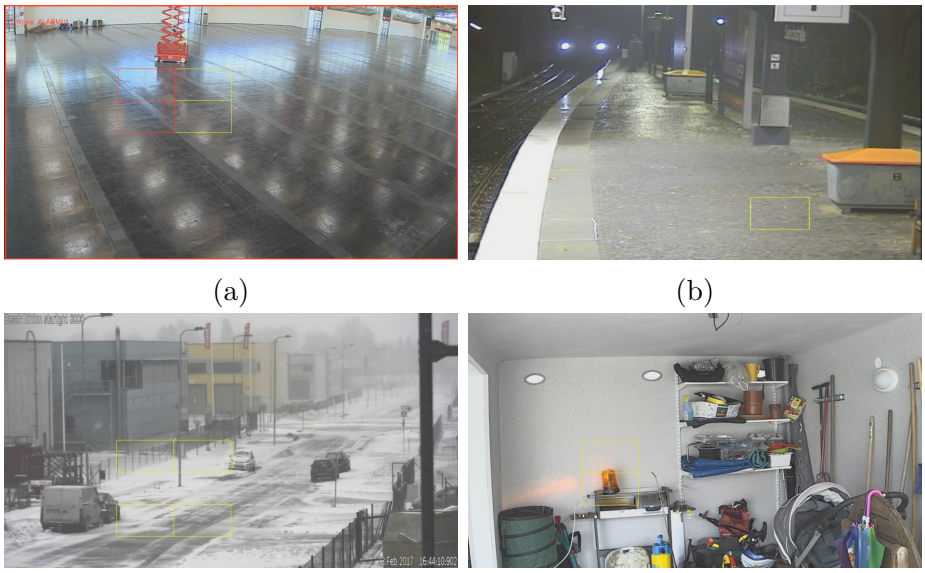


Figure 5.23: Comparison of critical and not critical negative events considering meaningful distributions of negative types and environmental conditions. Again the percentage values are given in relation to the number of events in each column. For example 31% of the not critical negative events and 42% of the critical negative events have the UE property.

All these critical events (except the blinking lights) have locally similar behavior to smoke, i.e., if a cell containing the critical event is cropped, not even a human



(a) (b)

(c) (d)

Figure 5.24: Examples for negative scenarios, which are critical. Yellow markers are pre-alarms (0.02 <FPR≤ 0.10), and red markers are smoke alarms (FPR≤ 0.02). (a) A slowly rising hydraulic ramp. The reflection on the sparkling ground causes a false alarm. (b) Slowly passing train towards the camera. The reflection of headlights causes the pre-alarm. (c) Snowing and windy. The event causing a pre-alarm is a snowdrift. (d) Turning blinking light. The similarity to flames causes a pre-alarm.

can decide for sure, if it is smoke or not. Only when considering the context of the event, a reliable decision can be made. For example, if in Figure 5.24c the part of the street containing the snowdrift is cropped, it looks like smoke, but considering the weather, which can be derived from the context, one can be sure that this is non smoke.

All proposed architectures can recognize the context, since the spatial receptive fields cover the whole image, but the dataset is not comprehensive enough, that the algorithm learns such complex correlations. For example, there is no sequence containing snow in the training set.

Dust or steam are not investigated as negative events in this thesis. However, using contextual information is –to the author's suggestion– the only way to distinguish them from smoke.

Chapter 6

Reduction of Model Complexity

So far, several single frame and temporal DL approaches are applied to VSD, and they are compared due to their performance in the requirements derived measure, *Detection Speed*. The architectures used for feature extraction are SotA in several public classification challenges. What is completely left out in the current analysis, is the complexity of the models.

VSD is a real-time problem, i.e., all calculations for the alarm decision have to be done in real-time. The real-time ability depends on the hardware and the model complexity, e.g., on-edge-devices usually not have the computing capability like a desktop PC or a server. Normally, a reduction of model complexity leads to some decrease in performance.

In this chapter, the complexity of the proposed models is analyzed and compared. Furthermore, several approaches are investigated to decrease complexity while maintaining performance.

In Section 6.1, the models from the last chapter are compared due to their complexity. Afterward, a custom architecture based on ConvLSTMs is proposed in 6.2, which decreases the model complexity drastically, but also the performance significantly decreases. In Section 6.3 a SotA CNN architecture, called *MobileNetV2*, to reduce the model complexity is investigated, and it is shown, that the performance only slightly decreases by using $< 50\%$ of the *InceptionV1* complexity.

3D-CNN complexity reduction is analyzed in Section 6.4. Firstly a SotA method is applied, which maintains the *i3D* performance by 64% complexity reduction. Secondly, the inflation concept of the *InceptionV1* architecture to *i3D* is applied to

yield an inflated *MobileNetV2* 3D architecture, which delivers better performance than CNN+LSTM, while reducing the model complexity to $< 1\%$ of the *i3D*.
From now on, only Diff input is considered because it shows by far the best results for single frame and temporal DL architectures.

6.1 Complexity Analysis of Proposed Models

With the increasing success of DL models, the question arises, how to apply them in applications, in which normally hardware restrictions exist. Therefore the goal of DL research is not only to increase performance but also to design efficient models with less complexity. One way to describe the model complexity is the number of parameters, which is an indicator of how "big" a model is but not how much calculations are necessary for one forward pass of an image through the network.

6.1.1 Definition of Model Complexity

According to [37] a suitable measure for model complexity is the number of multiplications and accumulations (MACs), which are needed for one forward pass. MACs are the dominant operation using neural networks since dot products can describe convolutions. Calculations needed for non-linearities are negligible.
The MACs can be calculated layer-wise and summed up over the whole network. They do not only depend on the architecture, but also on the input image size and the sequence length, since for the VSD use case the samples are defined as sequences. The image size is always $256 \times 256 \times 3$, but the sequence length varies. Therefore MACs per frame are used for model complexity comparison. It is assumed that the input sequence has a length of T. Now it is described how the MACs are calculated for different layers.

MACs of 2D Convolutional Layers

Since 2D convolutional layers are independent of time, the MACs per frame can directly be computed for each frame. Let $K \in \mathbb{R}^{k \times k \times c}$ be the kernel size, $I \in \mathbb{R}^{h \times w \times c}$ be the input and $O \in \mathbb{R}^{h' \times w' \times c'}$ the output of the layer. Note that h, w can be reduced

to h', w' using strides> 1. When ignoring padding effects, it follows

$$\text{MACs per frame} \approx k^2 h' w' c c'. \tag{6.1.1}$$

MACs of 3D Convolutional Layers

Since the MACs of 3D convolutions are dependent on the sequence length, the MACs have to be normalized by the number of input frames T. Let $K \in \mathbb{R}^{t \times k \times k \times c}$ be the kernel size, $I \in \mathbb{R}^{T' \times h \times w \times c}$ be the input and $O \in \mathbb{R}^{T'' \times h' \times w' \times c'}$ the output of the layer. When ignoring padding effects, it follows

$$\text{MACs per frame} \approx t k^2 T'' h' w' c c' / T. \tag{6.1.2}$$

Note that the input of the first layer is the number of input frames. Therefore the expression is divided by T.

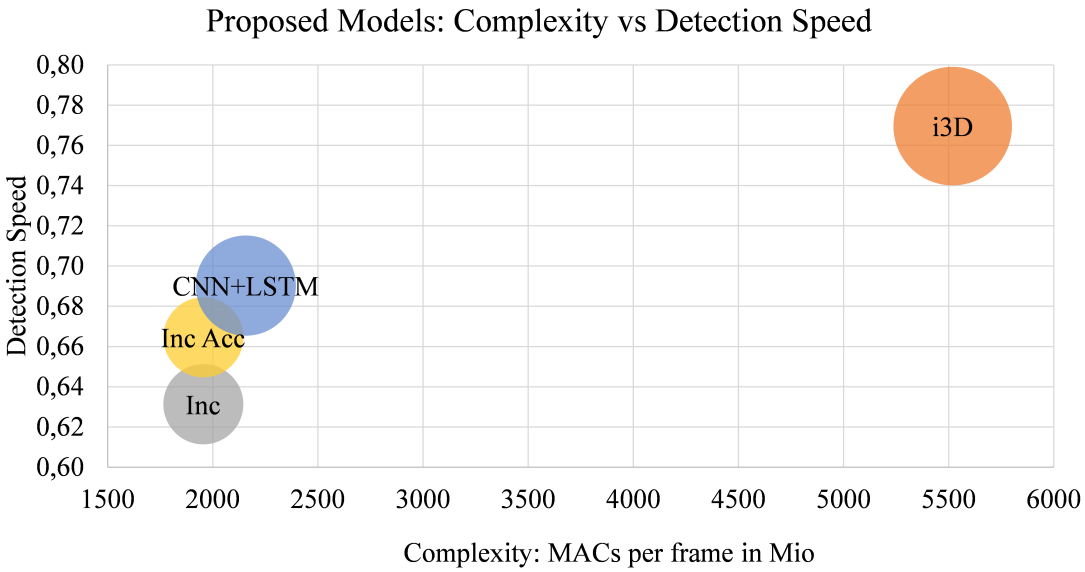
MACs of ConvLSTMs

Since there is no reduction in the temporal component by ConvLSTM, only one frame is considered when calculating the MACs per frame. Let $K \in \mathbb{R}^{k \times k \times c}$ be the kernel size, $I \in \mathbb{R}^{h \times w \times c}$ be the input and $O \in \mathbb{R}^{h' \times w' \times c'}$ the output of the layer. The calculation of the activations and the element-wise products and additions are negligible. When ignoring padding effects, it follows

$$\text{MACs per frame} \approx 8 k^2 h' w' c c'. \tag{6.1.3}$$

6.1.2 Complexity Comparison of Proposed Models

Figure 6.1 summarizes the resulting model complexities against the performance of the architectures proposed so far in a blob diagram. Since the recurrent accumulation operation is neglegible, the *InceptionV1* and the Acc approach have the same complexity and number of parameters. Whereas the CNN+LSTM approach has 57% more parameters, but only 10% more MACs are necessary for one forward pass



Model	#Params in Mio	MACs/frame in Mio	$D_{\text{Speed}}(0.02)$
Inc	5.59	1955	0.6313
Inc Acc	5.59	1955	0.6646
CNN+LSTM	8.76	2157	0.6903
i3D	12.28	5519	0.7696

Figure 6.1: Blob diagram, that shows the complexity of currently investigated models in MACs against the *Detection Speed* ($D_{\text{Speed}}(0.02)$). The area of the blob is proportional to the number of parameters. The table summarizes the results.

since the LSTM is applied on small input resolution. The *i3D* has by far the biggest complexity: 2.6 times the CNN+LSTM complexity and 1.4 times more parameters, since it completely consists of 3D convolutions.

6.2 DeepConvLSTM – A Custom Architecture

Now the goal is to reduce the model complexity while maintaining the performance. The first idea is a custom architecture, which can extract long and short term information at each layer. The investigation of CNN+LSTM shows, that extracting long term information at high level of spatial abstraction does have less benefit (see Section 4.6).

The question arises if it is possible to get better results by adding long term information at other levels of spatial abstraction. This could be done to the full extent by replacing convolutional layers in a CNN by ConvLSTMs (Figure 6.2). This

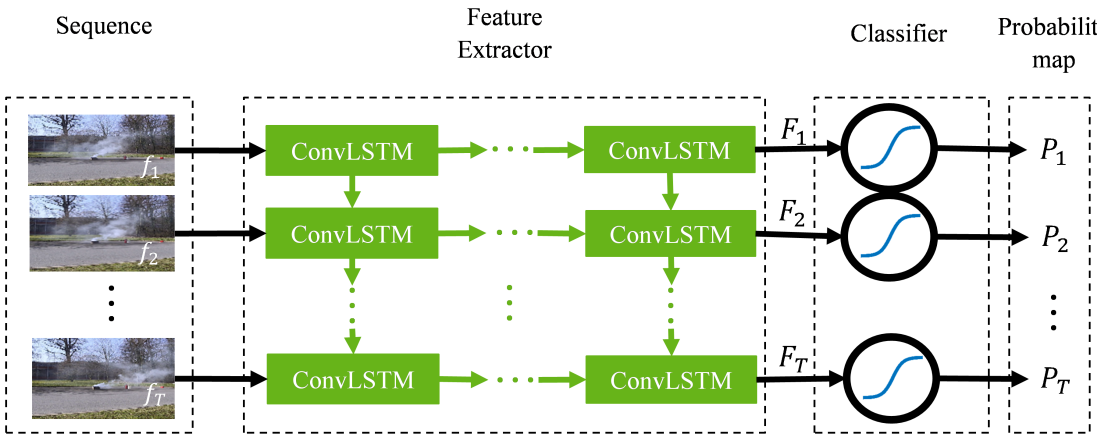


Figure 6.2: The arrows show the information flow of the DeepConvLSTM. The ConvLSTMs are stacked, so that the architecture has the ability to extract long and short term temporal information at each spatial abstraction level.

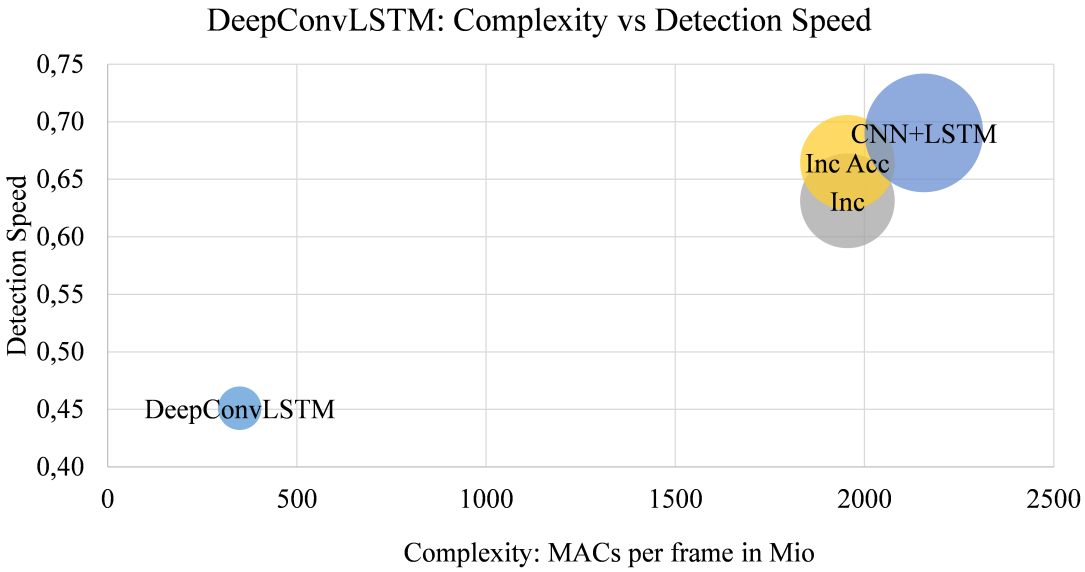
Layer	**type**	**kernel size/stride**	**output**	**max receptive field**
0	input	–	$T \times 256 \times 256 \times 3$	–
1	ConvLSTM	3×3	$T \times 256 \times 256 \times 4$	$T \times 3 \times 3$
2	3D Max-Pool	$1 \times 3 \times 3/(1,2,2)$	$T \times 128 \times 128 \times 4$	$T \times 5 \times 5$
3	ConvLSTM	3×3	$T \times 128 \times 128 \times 8$	$T \times 9 \times 9$
4	3D Max-Pool	$1 \times 3 \times 3/(1,2,2)$	$T \times 64 \times 64 \times 8$	$T \times 13 \times 13$
5	ConvLSTM	3×3	$T \times 64 \times 64 \times 16$	$T \times 21 \times 21$
6	3D Max-Pool	$1 \times 3 \times 3/(1,2,2)$	$T \times 32 \times 32 \times 16$	$T \times 29 \times 29$
7	ConvLSTM	3×3	$T \times 32 \times 32 \times 32$	$T \times 45 \times 45$
8	3D Max-Pool	$1 \times 3 \times 3/(1,2,2)$	$T \times 16 \times 16 \times 32$	$T \times 61 \times 61$
9	ConvLSTM	3×3	$T \times 16 \times 16 \times 64$	$T \times 93 \times 93$
10	3D Max-Pool	$1 \times 3 \times 3/(1,2,2)$	$T \times 8 \times 8 \times 64$	$T \times 125 \times 125$
11	ConvLSTM	3×3	$T \times 8 \times 8 \times 128$	$T \times 189 \times 189$
12	FC (sigmoid)	$1 \times 1 \times 1$	$T \times 8 \times 8 \times 1$	$T \times 189 \times 189$

Table 6.1: The DeepConvLSTM architecture. Behind each 3D convolutional layer are batch normalization layers. The activation used in the ConvLSTM are hard sigmoid and tanh.

concept is called DeepConvLSTM. The intuition of this architecture is, that it can extract long and short term information at each spatial abstraction level. In contrast 3D-CNNs can only extract short term information at each abstraction level and CNN+LSTMs can extract long and short term information only for the highest spatial abstraction level.

Furthermore, the architecture can learn more representative and simpler smoke information and therefore having less complexity.

A simple architectural concept is investigated (Table 6.1), which is basically an adaption of a *VGG* [33]. The network is trained with a sequence length of 100 frames each sample and a batch size of 5. The learning rate is decayed each epoch by a factor of 0.995 starting with 0.001. The test set is evaluated every 3 epochs. The training needs about 300 epochs for convergence. Figure 6.3 shows the result in the context of the SotA models from the last chapter. This simple architecture



Model	#Params in Mio	MACs/frame in Mio	$D_{\text{Speed}}(0.02)$
Inc	5.59	1955	0.6313
Inc Acc	5.59	1955	0.6646
CNN+LSTM	8.76	2157	0.6903
DeepConvLSTM	1.18	349	0.4510

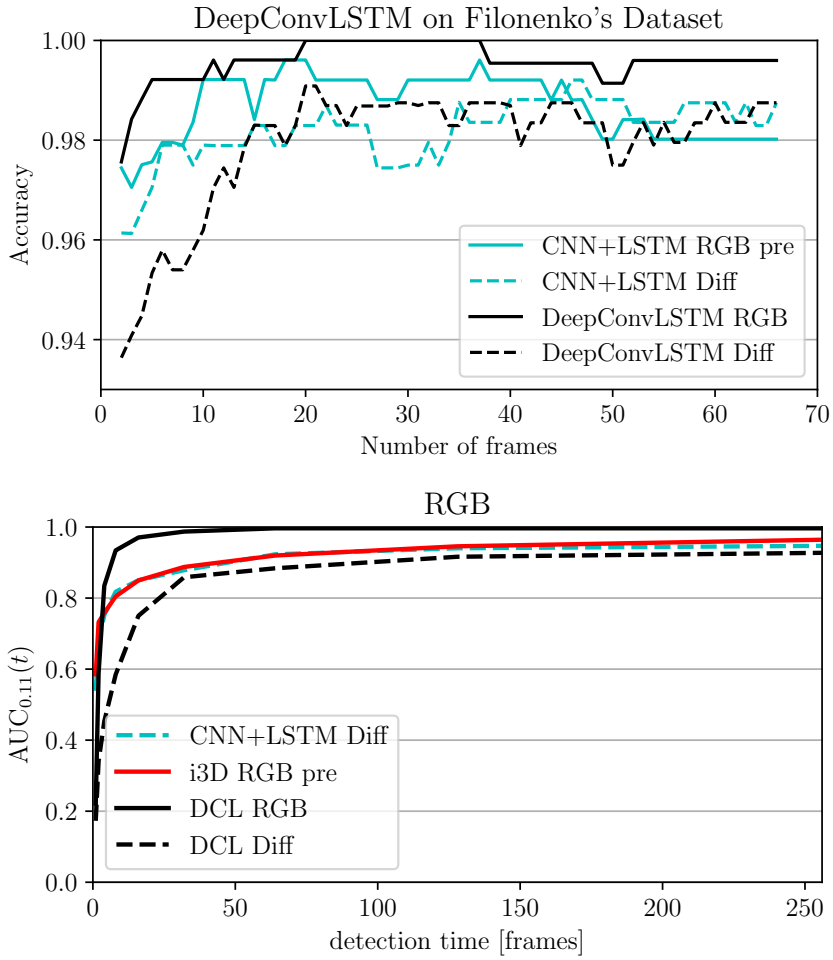
Figure 6.3: Blob diagram, that compares complexity and performance to the the other DL models.

delivers good results: It has only 16% model complexity and 13% parameters of the CNN+LSTM approach while maintaining 65% of the performance.

The DeepConvLSTM is also analyzed on the dataset of *Filonenko* (Figure 6.4). The DeepConvLSTM reaches second best results using Diff images as input, but despite the lack of pretrained weights for RGB all other approaches are outperformed by far in *Filonenko's* evaluation methods and in *Detection Speed*. The author's assumption, why the results on *Filonenko's* dataset are so good is that it is easier to learn long term information by a stack of LSTM than for only one LSTM layer.

One can conclude that a stack of ConvLSTM is a promising and simple approach

to reduce the model complexity and get good results compared to the baseline.



Model	**Input**	**Mean**	**Max**	$D_{\textbf{Speed}}(0.02)$
CNN+LSTM	Diff	0.9820	0.9921	0.9187
DeepConvLSTM	Diff	0.9798	0.9901	0.8785
CNN+LSTM pre	RGB	0.9868	0.9961	0.9231
i3D pre	RGB	0.9765	0.9921	0.9256
DeepConvLSTM	RGB	0.9958	1.0000	0.9836

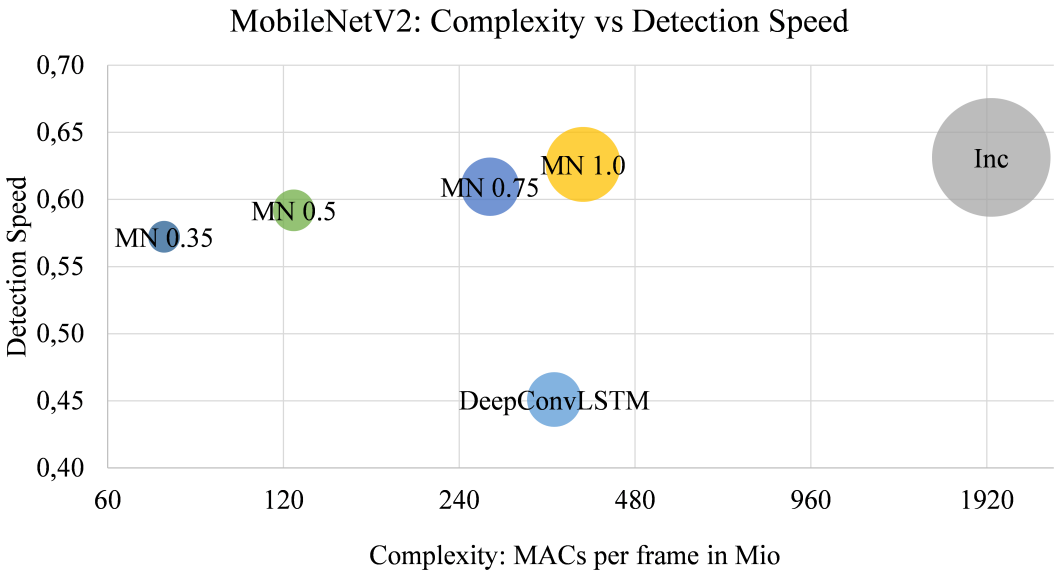
Figure 6.4: Evaluation of the DeepConvLSTM on the dataset of *Filonenko* using his measure and the measure presented in this thesis. For comparison the currently best models on *Filonenko's* dataset are also given. Diff and RGB input are tried as input. The table summarizes the results.

6.3 Complexity Reduction of CNNs

In literature, there are also some approaches to reduce the model complexity of CNNs while maintaining the performance. One SotA concept is introduced in [37],

the *MobileNetV2*. This architecture combines the full experience gathered by designing CNNs. However, the main idea to reduce the model complexity is depthwise separable convolutions, which consists of two parts. Depthwise convolutions are convolutions conducted for each input channel of a layer independently. Afterward, a 1×1 convolution is conducted to extract information between the channels. This reduces the MACs from $k^2h'w'cc'$ of a ordinary 2D convolution to $h'w'c(k^2+c')$. For example one has an input tensor of size $64 \times 64 \times 32$, 3×3 convolutional kernels with stride 1 and 64 output channels. A normal 2D convolutional layer needs 75M MACs, whereas a depthwise separable convolutional layer gets by with 10M MACs, but it has the opportunity to extract similar information.

The number of features extracted by each layer is controlled by $\alpha \in (0, \infty)$. On the



Model	**#Params in Mio**	**MACs/frame in Mio**	$D_{\textbf{Speed}}(0.02)$
Inc	5.59	1955	0.6313
DeepConvLSTM	1.18	349	0.4510
MN $\alpha = 1.00$	2.23	391	0.6261
MN $\alpha = 0.75$	1.36	271	0.6095
MN $\alpha = 0.50$	0.69	125	0.5919
MN $\alpha = 0.35$	0.40	75	0.5722

Figure 6.5: Blob diagram, that shows the complexity in MACs per frame against the *Detection Speed* ($D_{\text{Speed}}(0.02)$). The MACs per frame on the x-axis are scaled logarithmically. *MobileNetV2* (MN) for different α is compared to single frame concept Inception. Also, the DeepConvLSTM architecture is shown for model complexity comparison.

ImageNet dataset SotA results are reached for $\alpha = 0.35, 0.50, 0.75, 1.00, 1.40$. For

the purpose of this thesis $\alpha = 0.35, 0.50, 0.75, 1.00$ are trained in the same way like the Inception architecture, i.e. sequence length 1, batch size 162 and for learning rate decay factor 0.999 starting with 0.001. The results are shown in a blob diagram (Figure 6.5). Since the *MobileNetV2* should be applicated in the same way like the *InceptionV1*, i.e., as a backbone for the accumulation or CNN+LSTM approach, it is suitable only to compare it to the single frame results of *InceptionV1*: For $\alpha = 1.0$ the complexity is reduced to 20% and 99% of the performance is reached. For $\alpha = 0.35$, the model complexity is reduced to 4% and 91% of the performance is maintained, which is significantly better than the DeepConvLSTM.
Embedding the *MobileNetV2* architectures into the accumulation and CNN+LSTM approach should give the same improvement, but it is not investigated.

6.4 Complexity Reduction of 3D-CNNs

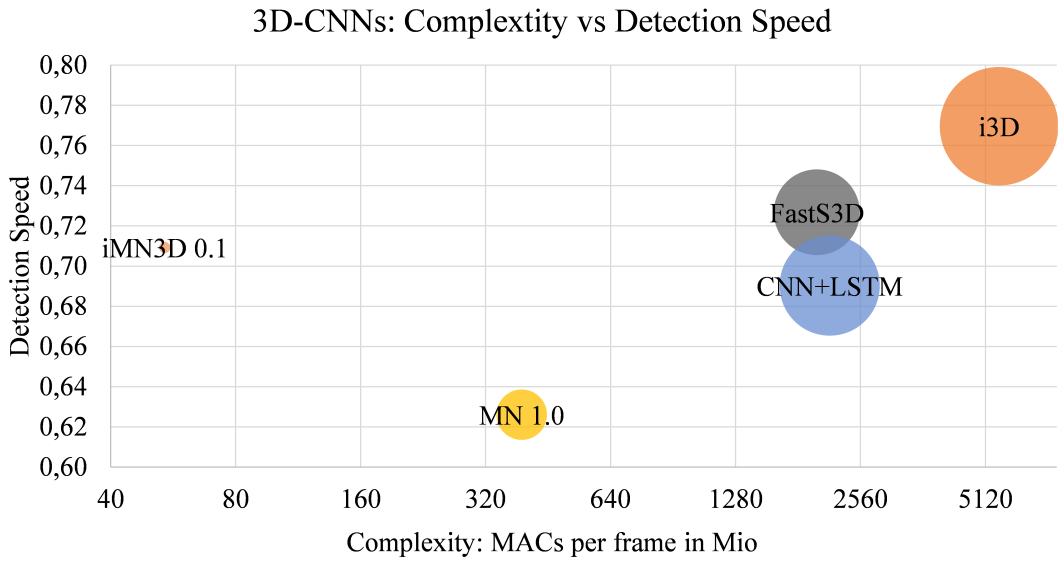
In Section 5.5, it is found that the *i3D*, a 3D-CNN architecture, delivers by far the best results for VSD, but it also has by far the highest model complexity. Now the question arises if there are 3D-CNN architectures, which deliver similar performance, but have less model complexity. Two approaches are considered, one from literature, which is mainly based on partly replacing the 3D convolutions by 2D convolutions, called *Fast S3D*, and the other one is a custom approach, which transforms the *MobileNetV2* into an inflated *MobileNetV2* 3D like it is done to yield the *i3D* by inflating the *InceptionV1* architecture. Both approaches deliver a performance worse than the *i3D* but better than the other approaches and especially the inflated *MobileNetV2* 3D architecture reduces the model complexity drastically.

6.4.1 Fast S3D

In [62] a simple idea to reduce the complexity of *i3D* is presented, where 3D convolutions are partly replaced by 2D convolutions. The strategy is successively replacing the 3D convolutions at the bottom and the top of the network. It turns out that replacing all 3D Inception modules by 2D modules at the bottom of the network except the last two is a sweet spot, i.e., a compromise between model complexity and performance.

Furthermore, the complexity is reduced by separating the temporal and the spatial part of a $3 \times 3 \times 3$ convolutional kernel into a stack of two kernels: $1 \times 3 \times 3$ and $3 \times 1 \times 1$. The resulting architecture using these two complexity reduction techniques is called *Fast S3D*.

The *Fast S3D* is similarly modified as the *i3D*, i.e., the temporal stride is set to 1, and padding is done, such that each frame yields cell-wise predictions for each frame. The spatial receptive field remains the same as Inception and *i3D*, but the temporal receptive field reduces to 7 frames. Figure 6.6 shows the result. The complexity is finally reduced to 36% of the *i3D* with 94% of the performance. Compared to the CNN+LSTM approach the complexity is 7% reduced and the performance 5% increased.



Model	**#Params in Mio**	**MACs/frame in Mio**	$D_{\textbf{Speed}}(0.02)$
CNN+LSTM	8.76	2157	0.6903
i3D	12.28	5519	0.7696
FastS3D	6.44	2009	0.7268
MN $\alpha = 1.0$	2.23	391	0.6261
iMN 3D $\alpha = 0.1$	0.11	54	0.7093

Figure 6.6: Blob diagram, that shows the complexity in MACs per frame against the *Detection Speed* ($D_{\text{Speed}}(0.02)$). The MACs per frame on the x-axis are scaled logarithmically. The main focus is on the 3D-CNN architectures *Fast S3D* and Inflated *MobileNetV2* 3D with $\alpha = 0.1$ (iMN3D 0.10).

6.4.2 Inflated MobileNetV2 3D

The *Fast S3D* has an *InceptionV1* like model complexity with *i3D* like performance. Compared to the *MobileNetV2* architectures, the complexity of this is still very high. The question arises if it is possible to get an *i3D* like performing architecture with *MobileNetV2* like model complexity.

The idea is to inflate the *MobileNetV2* to a 3D-CNN in the same way as the *i3D* is constructed from the *InceptionV1*.

Inflation concretely means that the 2D convolutions and the 2D depthwise separable convolutions have to be replaced by 3D counterparts, which is a straightforward extension. The temporal strides are set to 1, and padding is conducted, such that each frame yields cell-wise predictions. The resulting architecture is called Inflated *MobileNetV2* 3D. The spatial receptive field is the same as for the *MobileNetV2*, and the temporal receptive field is 37 frames.

Since there is no optimized version of depthwise 3D convolutions in *Tensorflow*, a custom version is implemented, which iterates over the input channels and is therefore slow on the GPU. Due to this only one small $\alpha = 0.1$ is tested (Figure 6.6). The resulting architecture only has a model complexity of 1% compared to the *i3D*, but 92% of the performance. Compared to the *Fast S3D*, it has 3% of complexity and maintains 98% of the *Detection Speed*. These investigations show that 3D-CNNs not only perform best for VSD. 3D-CNNs are also the best DL method to reduce the model complexity significantly while maintaining excellent performance.

Chapter 7

Conclusion

Firstly, the story of the proposed thesis is summarized in section 7.1. Afterward, the main results are emphasized in section 7.2. Finally, the main open questions, which arise during the research for this thesis, are given in section 7.3.

7.1 Summary

The goal in this thesis is to investigate, if and which DL methods can improve algorithms in the area of VSD. The key role of this investigation is data preparation and the definition of a measure for comparison. The investigations are done on an internal dataset of *Bosch Building Technologies GmbH*, which consists of about 1000 smoke and 1200 negative sequences.

These data are carefully structured into training and test sets. The key requirements for doing so are that each set is representative, and the sets are independent of each other. Representative is assured by constructing the sets, such that smoke types and environmental conditions have the same distribution. Independence is approximated by choosing disjoint recording locations for the sets.

Smoke is labeled with bounding boxes in the training sequences and by frame in the test sequences, which is a compromise between effort and labeling accuracy.

The different VSD algorithms are compared on the test set using a measure named *Detection Speed*, which takes values in the range from 0 to 1, the higher the better. This measure represents the ability of a VSD algorithm to detect smoke as fast as possible within a maximal required detection time of 90s.

Afterward, a VSD algorithm using classic CV methods is analyzed. The key idea of this algorithm is to accumulate the optical flow to detect smoke typical moving behavior over time. This algorithm reaches a *Detection Speed* of 0.1814, which leaves much space for improvement using DL approaches.

The training problem for the DL approaches is defined as a cell-wise classification problem: The input is a sequence of frames, and each frame is divided into a rough grid. For each cell in the grid and each frame, the DL algorithm predicts the probability that the cell contains smoke. To predict smoke in the current frame, the models only use information of previous frames in the sequence.

The investigation starts with the analysis of a CNN architecture, the *InceptionV1*. The first shot is using RGB images as input and train the network from scratch. The result is a *Detection Speed* of 0.1076, which is worse than the classic algorithm. When using transfer learning with pretrained weights on ImageNet, the result improves to 0.2670, which is significantly better than the baseline. The disappointing result without pretraining shows that the data set is not comprehensive enough to learn static, single frame-based, features, which are characteristic of smoke. However, the good result with pretraining indicates that this characteristic information is in the dataset.

In classic VSD algorithms, temporal information is highly relevant to detect smoke, which also follows the human intuition. One way to use temporal information in a CNN is to put this information in the input. This is done by using Flow and Diff images, which leads to a significant improvement: 0.2252 for Flow and 0.6313 for Diff, which is impressive since this information was only taken from two consecutive frames within 333ms. One also observes that using Diff input nearly no overfitting effects occurred in contrast to the RGB input.

Following the intuition, that smoke has the expanding property and frames, which are covered by smoke, usually remain covered for the next frames, a simple handcrafted extension of the CNN is introduced: The smoke probabilities of each cell are temporally accumulated over time by a first-order low pass filter. This stabilizes the CNN output and leads to a performance of 0.6646 for Diff input.

The next step is to investigate DL methods, which can extract temporal information on their own. The combination of CNN and LSTM directly delivers an improvement

to: 0.3775 for RGB and 0.6903 for Diff.

The last investigated DL concept is 3D-CNNs. The architecture, which shows the best results in action recognition tasks, is the *i3D*. RGB and Flow weights, which are pretrained on *Kinetics*, are available. The *i3D* shows the best performance when the same input modalities are compared, 0.5650 for RGB and 0.5139 for Flow. Both results are worse than CNN with Diff input, but using Diff input for the *i3D* the result increases to 0.7696, the best performance reached over the whole thesis.

These DL models have high computing complexity and are not applicable when having hardware restrictions, especially because VSD is a problem, which has to be solved in real-time. The model complexity is measured in MACs per frame. Several approaches are investigated to reduce the model complexity while maintaining performance. Starting with a custom architecture, the DeepConvLSTM, which is a stack of ConvLSTMs, reduced the complexity to 16% of the *InceptionV1* architecture, but also the performance reduced to 68% of the CNN+LSTM approach. Then an approach from literature is tested, the *MobileNetV2*, which reduces the complexity down to 4%-20% of the *InceptionV1* architecture, but still maintains 91%-99% of the *InceptionV1* performance.

The *i3D* has the highest model complexity 2.6 times the complexity of CNN+LSTM but reaches by far the best results using 3D convolutions. An approach from literature to reduce the model complexity is investigated, the *Fast S3D*, which replaces the most 3D convolutions by 2D convolutions. With 36% computing complexity it reaches 94% of the *i3D* performance. Finally, the custom idea to inflate the 2D *MobileNetV2* to the 3D version is investigated. This 3D *MobileNetV2* only needs 1% computing complexity and meets 92% of the *i3D* performance, which is still better than the CNN+LSTM.

7.2 Results

Data preparation and structuring is crucial for applying DL methods to VSD. If it is done insufficiently, any result is vulnerable.

The proposed *Detection Speed* measure enables to evaluate and delimit VSD algorithms in the context of real surveillance scenarios.

It is shown that DL can close the gap to fully automatic VSD. Only smoke events, which are even hard to detect by humans and negative events, which need contextual information to be distinct from smoke, are critical.
Among all investigated DL methods the one with 3D convolutions work best and all approaches benefit from Diff input.
When meeting hardware restrictions, it is possible to reduce the model complexity. For all investigated approaches, the computing effort can significantly be reduced. The best compromise between complexity and performance is again obtained by utilizing 3D-CNNs.

7.3 Open Questions

During the investigations in this thesis also some questions occurred, which are not sufficiently answered yet.
It is found that 3D-CNNs perform better than RNNs even though RNNs could benefit from much more temporal information than 3D-CNNs. The assumption is that it is hard to teach RNNs long term information.
It is also found that Diff as inputs works much better than RGB even when using temporal DL approaches and pretraining on RGB is available. This is contradictory, since Diff information should be learnable from RGB input. The explanation for this observation is the tendency of fast overfitting when using RGB.
Nevertheless those effects needs a deeper insight in what the DL methods learn and how one can influence this to further improve the results in VSD. There is an own field of research dealing with such questions, which is called "explainable AI".
Another problem, which occurs, when investigating DL algorithms with research purposes is that there is no relevant public dataset available. Investigations done on internal datasets are hard to comprehend, suffer from less credibility and are not comparable to other authors. A sufficient public dataset with events created and labeled to the guidelines given in this thesis would be of high value for the whole VSD community. This includes smoke and negative events recorded in diverse recording locations.

References

[1] R. Campbell, "Fires in industrial and manufacturing properties," *National Fire Protection Association*, 2018.

[2] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, "An early fire-detection method based on image processing," *International Conference on Image Processing 2004*, vol. 3, pp. 1707–1710, 2004.

[3] T. Celik, H. Ozkaramanli, and H. Demirel, "Fire and smoke detection without sensor: Image processing based approach," *15th European Signal Processing Conference*, pp. 1794–1798, 2007.

[4] B. U. Töreyin, Y. Dedeolu, and A. E. Cetin, "Wavelet based real-time smoke detection in video," *13th European Signal Processing Conference*, pp. 1–4, 2005.

[5] B. U. Töreyin, Y. Dedeolu, and A. E. Cetin, "Contour based smoke detection in video using wavelets," *14th European Signal Processing Conference*, pp. 1–5, 2006.

[6] J. Gubbi, S. Marusic, and M. Palaniswami, "Smoke detection in video using wavelets and support vector machines smoke detection in video using wavelets and support vector machines," *Fire Safety Journal*, vol. 44, no. 8, pp. 1110 – 1115, 2009.

[7] P. Piccinini, S. Calderara, and R. Cucchiara, "Reliable smoke detection in the domains of image energy and color," *15th IEEE International Conference on Image Processing*, pp. 1376–1379, 2008.

[8] Z. Xu and J. Xu, "Automatic fire smoke detection based on image visual features," *International Conference on Computational Intelligence and Security Workshops*, pp. 316–319, 2007.

[9] S. Calderara, P. Piccinini, and R. Cucchiara, "Smoke detection in video surveillance: A mog model in the wavelet domain," *Lecture Notes in Computer Science*, vol. 5008, pp. 119–128, 2008.

[10] Y. Chunyu, Z. Yongming, F. Jun, and W. Jinjun, "Texture analysis of smoke for real-time fire detection," *Second International Workshop on Computer Science and Engineering*, vol. 2, pp. 511–515, 2009.

[11] R. Gonzalez-Gonzalez, V. Alarcon-Aquino, R. Rosas-Romero, O. Starostenko, J. Rodriguez-Asomoza, and J. M. Ramirez-Cortes, "Wavelet-based smoke detection in outdoor video sequences," *53rd IEEE International Midwest Symposium on Circuits and Systems*, pp. 383–387, 2010.

[12] F. Yuan, "Video-based smoke detection with histogram sequence of lbp and lbpv pyramids," *Fire Safety Journal*, vol. 46, no. 3, pp. 132 – 139, 2011.

[13] F. Yuan, Z. Fang, S. Wu, Y. Yang, and Y. Fang, "Real-time image smoke detection using staircase searching-based dual threshold adaboost and dynamic analysis," *IET Image Processing*, vol. 9, no. 10, pp. 849–856, 2015.

[14] C.-Y. Lee, C.-T. Lin, C.-T. Hong, and M.-T. Su, "Smoke detection using spatial and temporal analyses," *International Journal of Innovative Computing, Information and Control*, vol. 8, pp. 4749–4770, 2011.

[15] W. Ye, J. Zhao, S. Wang, Y. Wang, D. Zhang, and Z. Yuan, "Dynamic texture based smoke detection using surfacelet transform and hmt model," *Fire Safety Journal*, vol. 73, pp. 91–101, 2015.

[16] K. Dimitropoulos, P. Barmpoutis, and N. Grammalidis, "Higher order linear dynamical systems for smoke detection in video surveillance applications.," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 27, no. 5, pp. 1143 – 1154, 2017.

[17] F. Yuan, "A fast accumulative motion orientation model based on integral image for video smoke detection," *Pattern Recognition Letters*, vol. 29, no. 7, pp. 925 – 932, 2008.

[18] C. Yu, J. Fang, J. Wang, and Y. Zhang, "Video fire smoke detection using motion and color features," *International Conference on Computational Intelligence and Security Workshops*, vol. 46, pp. 651–663, 2010.

[19] D.-K. Kim and Y.-F. Wang, "Smoke detection in video," *WRI World Congress on Computer Science and Information Engineering*, vol. 5, pp. 759–763, 2009.

[20] T. Truong and J.-M. Kim, "An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems," *Fire Safety Journal*, vol. 46, no. 5, pp. 276 – 282, 2011.

[21] T. Celik, H. Demirel, and H. Ozkaramanli, "Automatic fire detection in video sequences," *Proceedings of European Signal Processing Conference*, 2006.

[22] I. Kolesov, P. Karasev, A. Tannenbaum, and E. Haber, "Fire and smoke detection in video with optimal mass transport based optical flow and neural networks," *17th IEEE International Conference on Image Processing*, pp. 761–764, 2010.

[23] C. Yu, Z. Mei, and X. Zhang, "A real-time video fire flame and smoke detection algorithm," *9th Asia-Oceania Symposium on Fire Science and Technology*, vol. 62, pp. 891 – 898, 2013.

[24] J. Yang, F. Chen, and W. Zhang, "Visual-based smoke detection using support vector machine," *Fourth International Conference on Natural Computation*, vol. 4, pp. 301–305, 2008.

[25] Y. Jia, J. Yuan, J. Wang, J. Fang, Q. Zhang, and Y. Zhang, "A saliency-based method for early smoke detection in video sequences," *Fire Technology*, vol. 52, no. 5, pp. 1271–1292, 2016.

[26] S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," *IECON*

2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society Industrial Electronics Society, 2016.

[27] A. Filonenko, L. Kurnianggoro, and K.-H. Jo, "Comparative study of modern convolutional neural networks for smoke detection on image data.," *10th International Conference on Human System Interactions (HSI)*, p. 64, 2017.

[28] Y. Hu and X. Lu, "Real-time video fire smoke detection by utilizing spatial-temporal convnet features," *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29283–29301, 2018.

[29] A. Filonenko, L. Kurnianggoro, and K.-H. Jo, "Smoke detection on video sequences using convolutional and recurrent neural networks," *Computational Collective Intelligence*, pp. 558–566, 2017.

[30] G. Xu, Y. Zhang, Q. Zhang, G. Lin, and J. Wang, "Deep domain adaptation based video smoke detection using synthetic smoke images," *Fire Safety Journal*, 2017.

[31] A. E. Cetin, K. Dimitropoulos, B. Gouverneur, N. Grammalidis, O. Günay, Y. H. Habiboğlu, B. U. Töreyin, and S. Verstockt, "Video fire detection review," *Digital Signal Processing*, vol. 23, no. 6, pp. 1827 – 1843, 2013.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *CVPR09*, 2009.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *The 3rd International Conference on Learning Representations*, 2015.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[36] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1800–1807, 2017.

[37] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[38] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[39] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *ArXiv*, vol. abs/1705.06950, 2017.

[40] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," *ArXiv*, vol. abs/1808.01340, 2018.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 1997.

[42] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems 28*, 2015.

[43] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, 2010.

[44] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," *Proceedings of the 31st International Conference on Machine Learning*, pp. 647–655, 2014.

[45] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.

[46] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[47] R. B. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[49] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[50] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[51] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *NIPS*, 2014.

[52] A. Wellhausen and A. Stadler, "A smoke type classification concept for video smoke detection," *Aube/Supdet*, 2017.

[53] A. Wellhausen, A. Stadler, and F. Hoppe, "Visual smoke density measurement for video smoke detection," *Aube/Supdet*, 2017.

[54] A. Stadler, "Video basierte rauchdetektion durch mehrfache bewegungsschätzung und anpassungen für geringe rauchdichten," *Technische Universität München*, 2017.

[55] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, 2015.

[56] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proceedings of the 32nd International*

Conference on International Conference on Machine Learning, vol. 37, pp. 448–456, 2015.

[57] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[58] F. Chollet *et al.*, "Keras," *https://keras.io*, 2015.

[59] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," *http://tensorflow.org/*, 2015.

[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[61] J. Snchez Prez, E. Meinhardt-Llopis, and G. Facciolo, "Tv-l1 optical flow estimation," *Image Processing On Line*, vol. 3, pp. 137–150, 2013.

[62] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning for video understanding," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

DOI: 10.17185/duepublico/75984
URN: urn:nbn:de:hbz:465-20220624-104449-0