Wildfire Smoke Detection using Convolutional Neural Networks

Master Thesis

Simon Philipp Hohberg

09/20/2015

Freie Universität Berlin
Fachbereich Mathematik und Informatik

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Planetenforschung Berlin

Adviser
Dipl.-Inf. Barbara Haupt

Superviser
Prof. Dr. Raúl Rojas

Eidesstattliche Erklärung

Ich versichere, die Masterarbeit selbstständig und lediglich unter Benutzung der angegebenen Quellen und Hilfsmittel verfasst zu haben.

Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Berlin, 20. September 2015

Abstract

Wildfires pose a threat to nature and human health especially in the age of global warming. Their early detection is key to an effective fighting, because once a wildfire reaches a certain size it can be hardly controlled.

Consequently, automated wildfire detection systems have been developed to aid rangers in their work to prevent wildfire hazards. One of the most successful systems was developed by the German aerospace center DLR (Deutsches Zentrum für Luft- und Raumfahrt) and relies on sophisticated hand-crafted features. The algorithm behind this system is called *fshell*.

Convolutional neural networks (CNN) are specialized artificial neural networks that are the state-of-the-art for image recognition tasks. It is therefore investigated if these trainable models can be applied to the wildfire detection problem to improve the performance of existing systems. Besides CNNs that use spatial features only, C3D networks that can also extract temporal features are considered. To tackle the detection of smoke in a larger image, a selective search variant was developed, that preselects regions of observable motion, which are then classified by the actual models.

The results clearly show that the trained networks learned to distinguish wildfire smoke from other objects, although they do not reach the *fshell*'s performance. However, *fshell* makes use of additional prior knowledge whereas the CNNs rely on image data only. The results also show that C3D networks are the best performing single models, suggesting that the use of temporal features is important for accurate detection.

Contents

1 Introduction

Wildfires claim the lives of many people every year and cause huge environmental and economical damages. Many great wildfire disasters can be found in history like the Landes Forest wildfire in 1949 in France that claimed 82 lives and destroyed 140,000 ha of forest which equals to more than the area of Berlin.[56] But also in more recent years wildfires can get still out of hand like 2003 in Portugal, where wildfires in one of the hottest summer seasons destroyed 10% of the overall forest area in Portugal (215,000 ha) and killed 18 people.

In the age of global warming, the reduction of CO_2 emissions is crucial to prevent to irreversibly change world's climate. Here wildfires are a factor not to be ignored, since 20% of the CO_2 in the atmosphere comes from wildfires.[38] Even worse, global warming and wildfires mutually reinforce each other. The increased average temperature, due to global warming leads to increased temperature extremes and longer dry periods, which increases the risk of wildfires. An increased number of wildfires, however, leads to higher CO_2 emissions fueling global warming.

Consequently, the prevention, early detection and suppression of wildfires remains a very important task. Automated wildfire detection systems help spotting fires early allowing firefighters suppressing them before they are getting out of control. In addition, those systems can be deployed at large scale, in uninhabited regions and backwoods, where a traditional fire watch employing rangers could be hardly realized, for an almost complete forest monitoring.

1.1 The FireWatch Wildfire Detection System

Many different algorithmic approaches exist that try to tackle the problem of automated wildfire detection based on images taken from surveillance cameras. One of the most successful marketed fire detection systems called *FireWatch* was developed by the DLR. It relies completely on handcrafted features to detect smoke. Although its performance is probably one of the best of the available smoke detection systems, there is still room for improvement.

The Bushfire Cooperative Research Center in Australia evaluated three wildfire detection systems, including *FireWatch*. One of the systems did not report alerts automatically. The other two systems, *FireWatch* and *ForestWatch*, detected only

a small fraction of the wildfires, whereas *FireWatch* performed better than *ForestWatch*. Depending on the fire source the systems detected only 25% of the fires in the worst and 55% in the best case.[54]

The *FireWatch* system was originally developed to be deployed in the forest of the German state Brandenburg. Brandenburg is covered with about 1.1 hectare of wood (37% of the area) and is one of the German states with the highest forest coverage.[25] A great fraction of it is coniferous forest (70% pines) on sandy ground.[16] As a result the risk for wildfires especially in hot summer periods is very high. Historically, a fire watch system with watch towers was therefore build to discover wildfires. Spotters would sit on these towers to look out for smoke emissions and alert fire fighters, so that wildfires could be suppressed as early as possible. This is of course a very tedious, but also responsible task for the watchers. That is why attempts were made to create an automated system that can take over this task.

The European commission classifies forest in the EU according to the risk of wildfires. Brandenburg received the highest class in this ranking on the same level with south France and south Spain. Consequently, the EU financed a project that aimed at developing an automated early wildfire detection system in 1997. The DLR was in charge to develop technology that is able to reliably detect wildfires as early as possible. This was due to the DLR's expertise in computer vision tasks and the availability of the high quality camera system ROsetta Lander Imaging System (ROLIS) developed by the DLR, which is the same camera system deployed on the Rosetta space probe.

In addition to providing the initial camera system, the DLR developed an algorithm called *fshell* that uses the images to automatically detect wildfire smoke. The camera system in combination with the *fshell* algorithm was successfully tested in 1998 and finally won against competing systems in 2001 receiving financial support for series production. The commercial marketing of the system under the name *FireWatch* was then done by the IQ Wireless GmbH licensing the DLR system.

Over the years the system was further improved and deployed in more and more states within Germany and also in countries all over the world like Estonia, Cyprus, Mexico, Spain, USA and Kazakhstan.[37] Today, the system covers the whole forest in Brandenburg protecting it from wildfires and is considered a great success.

The system as it is deployed today is made up of camera systems developed by IQ Wireless GmbH themselves, which are set up on fire watchtowers. The cameras take gray-scale pictures in the visible light range at day and in the near infrared range at night. There is a single camera on each tower that rotates in 10° steps within 4 to 10 minutes and takes three images at an interval of ten seconds at every position. The *fshell* algorithm is deployed on each tower to locally evaluate the images. When the system detects smoke an alarm is sent via a wireless connection to a central office where rangers further check generated alerts as can be seen in Figure 1.1. The ranger then takes the final decision whether there really might be a wildfire or if it was a false alert.[37]

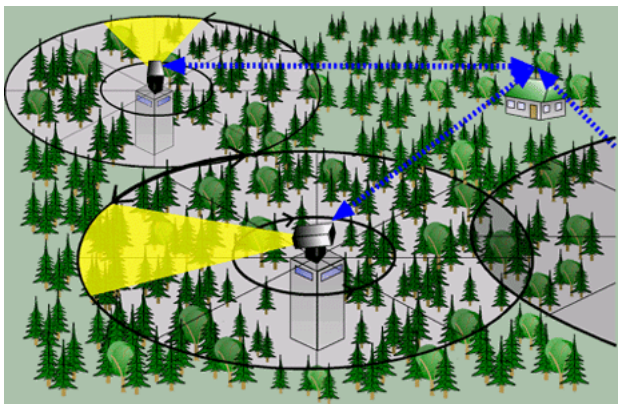

Figure 1.1: Overview of the *FireWatch* system setup [37]

1.2 Wildfire Smoke Detection Problem

Image recognition is one field in computer vision. Generally, it is the task of identifying one or many objects in an image. Different sub-categories of this problem are usually distinguished: image classification, object detection and image segmentation.

Assigning one of a set of predefined class labels to a whole image is considered *image classification*. This presumes that there is only a single dominant object in each presented image.

Object detection can be seen as generalization of image classification. Presented images can contain an arbitrary number of objects that have to be identified. This includes localizing each object in the image as well.

In contrast to object detection which classifies regions in a larger image, *image segmentation* is the task of labeling each pixel in a given image, so that it is segmented into non-overlapping patches belonging to different classes. Formally image segmentation can be defined as the problem of partitioning a given image into patches so that the union of all patches is the original image and each patch does not overlap with any other patch.[69]

The problem to be solved is the detection and localization of wildfire smoke in an image sequence, i.e. labeling a sequence to contain smoke or not. A sequence contains three frames that were taken at a ten seconds interval. This task is a classical objection detection task with two classes. One class for the wildfire smoke and another class for anything else not being smoke. This classification is considered a binary classification, since the problem requires only a yes/no or present/not present decision.

Because a CNN has a fixed input size the whole problem cannot be solved at once by training a CNN only. In fact, the task is divided into two parts. The first relies in deciding how to apply the model to a larger image to localize the objects

the model was trained on. The second part then consist of training a CNN on a dataset that contains positive and negative examples appropriately resized to be input to the CNN that were selected according to the approach chosen for the first part. Obviously these two parts dependent on each other, because the selection of examples depends on how to localize the objects.

Here one major difficulty becomes clear already. The *not smoke* class is much larger than the *smoke* class and the question arises how to select negative examples for training. This issue will be further addressed in section 5.3.

Since a sequence contains three images, another issue that needs to be addressed is, how to evaluate a sequence given a CNN that works on single images only. Here the selected solution is to evaluate a given region on each image and taking the average over all three predictions. For models that also use the temporal dimension, this approach does not apply, because these models evaluate a given region directly for the whole sequence.

1.3 Convolutional Neural Networks

Although many publications on wildfire smoke detection algorithms exist, yet there have been no efforts to apply deep learning approaches to the problem.

CNNs are a biologically inspired specialized version of artificial neural networks, developed specifically for vision tasks. Artificial neural networks are a mathematical model that can be trained to represent arbitrary mathematical functions.[63] CNNs as descendants of artificial neural networks belong to the class of deep learning algorithms, that can, in the same way, learn complex representations from large amounts of image data. The word deep, refers to the hierarchical architecture of these models, i.e. these algorithms extract less complex features at stages closer to the input and then combine those features to form more and more complex representations.

Although the basic concepts of CNNs are known since 1980, they did not receive a lot of attention until the last few years. Due to theoretical advances and the increased availability of computational resources and larger amounts of data, CNNs are now considered state-of-the-art in many vision tasks. Because of the great performance of CNNs in the field of computer vision they are a very promising candidate for improving wildfire detection based on image data.

1.4 Structure of this thesis

The structure of this thesis is as follows: In chapter 2 I will introduce artificial neural networks and describe their theoretical foundations. In chapter 3 I will then

give an overview of the state-of-the-art in smoke detection and image recognition using CNNs. The data that was used for training and validation is described in chapter 4. How the data is preprocessed and the detection problem is approached is then explained in chapter 5, which further includes the description of the selected models.

chapter 6 sums up the results for all trained models and investigates what they learned. Finally, chapter 7 concludes the findings based on the evaluation of the models.

2 Artificial Neural Networks

Artificial neural networks are the attempt of mimicking how biological nervous systems like the human brain work. This chapter introduces the fundamentals of the biological model of neural networks and shows how it inspired the development of artificial neural networks.

Furthermore, the mathematical model behind artificial neural networks and CNNs is described.

2.1 The Biological Model

A great step in understanding the structure and functions of the nervous system was the formulation of the neuron doctrine by Cajal.[17] It describes the discovery that the whole brain is made from elementary signaling units, the neurons. These neurons all share the same components and basic functionality, nonetheless many different types of neurons with slight differences exist. Figure 2.1 shows five types of neurons, each of them having the four base elements: cell body, dendrites, axon and synapse.

The synapses are the contact points between neurons, i.e. where output and input channels of neurons meet. The input channels through which a neuron receives signals from other neurons are called dendrites. Usually a neuron receives inputs from many other neurons so that there are many dendrites. Although a neuron always has only a single output channel, the axon, it can input signals to many other neurons as well, because the axon can develop branches at its end where dendrites of other neurons can connect via synapses.

Neurons being highly specialized cells, also own a cell body. Its purpose in the case of neurons is to provide the necessary chemicals that are used to transport the signals. As already implicitly stated, signals flow through a neuron only in a single direction from the dendrites through the cell body to the axon. This was also a discovery by Cajal.[43]

The development of artificial neural networks, however, is not only the result of advances in neuroscience but also in statistics and information theory. It can be argued that the artificial neuron is not just a copy of the concepts found in nature, but is the straight forward result when searching for the optimal discriminant function for

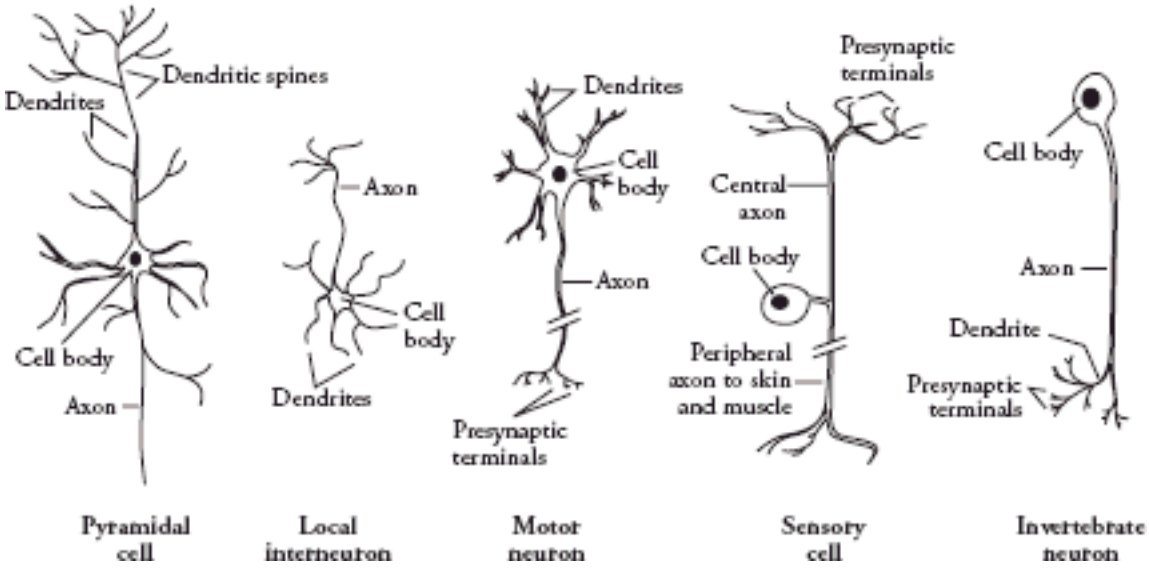

Figure 2.1: Different types of neurons, all sharing the same four basic components: cell body or soma, dendrites, axon and synapse. [21]

distinguishing two classes.[14] As a result, artificial neurons are indeed biologically inspired but abstract the functionality of a biological neuron very much.

Mathematically, artificial neural networks are just another computation model like the Turing machine and modern computers.[63] Consequently, they are equivalently powerful as these systems being able to implement any computable function, i.e. they are Turing complete, considering recurrent neural networks.[5]

From this computational perspective biological neural networks have some remarkable properties. Although biological neurons are slow compared to modern computer hardware – neuronal activity is measured in milliseconds whereas CPUs perform operations in nanoseconds – humans are able to perform complex tasks within a few hundreds milliseconds.[66] Their powerfulness can therefore only derive from a great number of replicated simple and rather slow units, whose specific structure and communication, i.e. the flow of information, determines the whole system's function. The high grade of parallelization and redundancy is key to the effectiveness of a nervous system building on these units.[63]

2.2 McCulloch Pitts Unit

In 1943 McCulloch and Pitts published their work on the computational capabilities of the human nervous system. For their experiments they implemented a simple neuron model and are thus considered the first for creating an artificial neural network.[55]

They used a very simple neuron model that is shown in Figure 2.2. The model has excitatory and inhibitory input channels representing the dendrites. The excitatory signals are summed and compared to a threshold. When the sum exceeds the threshold the unit fires. However, if only a single inhibitory signal is present the unit is retained from firing. This models the soma's function in the biological neuron based on the findings about the human brain at that time. The axon is implemented as the possibility of an output signal to be input to other neurons again.

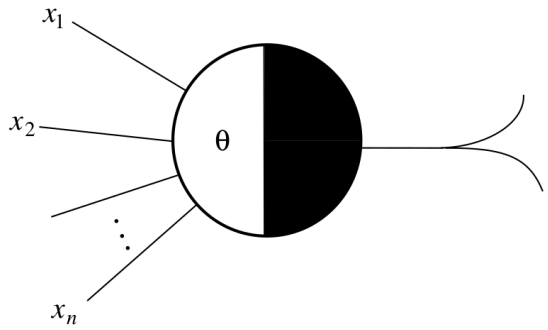

Figure 2.2: McCulloch-Pitts unit with inputs x_i and a threshold θ.[63]

In the model of McCulloch and Pitts only binary signals and multilayered acyclic networks of units were considered. Also there are no free parameters that can be adapted and thus no learning is performed. As a result such networks were restricted to only logical functions. Nonetheless, McCulloch and Pitts already found that networks of this type of units can compute any logical function.[55]

2.3 Perceptron

A more general version of the McCulloch-Pitts unit, called the *perceptron*, was introduced by Rosenblatt in 1957.[64] The major improvement was the introduction of weighted connections between units and a learning algorithm that adapts these weights so that the network is able to learn.[63] The actual goal of Rosenblatt was to create a more complex model for image recognition that used a single layer of *perceptrons*. Due to the fact that the model utilized only a single layer of neurons, the tasks that could be solved by Rosenblatt's model was still very limited as was analyzed by Minsky and Papert. They found that a single layer of *perceptrons* cannot implement the XOR function, because it can only implement functions that are linearly separable.[63]

Figure 2.3 shows the general definition of an artificial neuron. It covers both the *perceptron* model as defined by Rosenblatt and how artificial neurons are modeled today. The inputs x_i are aggregated by the function g. An activation function f is then applied on the result of g.

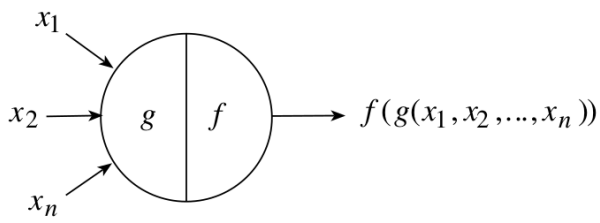

Figure 2.3: General structure of an artificial neuron.[63]

In the case of the *perceptron,* g is the summation and f is the threshold function. To be able to learn the threshold of an artificial neuron as well, Widrow established the idea of adding a constant bias term as an extra input to each unit.[76] Like all other inputs its connection has an associated weight, too. As a result, the activation function's threshold can be also learned during training.

2.4 Multilayer Perceptron

It was the discovery of the backpropagation algorithm that can be used to train networks with multiple layers of perceptron units (multilayer perceptron, MLP), which again increased interest in artificial neural networks. It was independently discovered multiple times.[63] The work of Werbos can be seen as the root of the backpropagation algorithm. However, the work of Rumelhart et al. in the Parallel Distributed Processing (PDP) group of the University of California also had a great influence in this area.

The backpropagation algorithm solves the problem of adapting the weights in each layer of a MLP minimizing a given error function that depends on the network's output.[63]

Assuming a network of units that use summation for aggregating inputs and an activation function f, the output o_j^l of unit j at layer l is the value of f applied to the weighted sum of all outputs from the units in the previous layer a_j^l as defined in Equation 2.1 and Equation 2.2. w_{ij}^l denotes the weight of the connection between unit i at layer $l-1$ and j at layer l. Further considering the mean squared error as error function as defined in Equation 2.1, o_i^L representing the output of unit i in the last layer of the network and t_i the target value, minimizing the error regarding a certain weight requires calculating the partial derivative of the error with respect to this weight.

$$o_j^l = f\left(a_j^l\right) \tag{2.1}$$
$$a_j^l = \sum_i w_{ij}^l o_i^{l-1} \tag{2.2}$$
$$E = \frac{1}{2}\sum_i \left\| o_i^L - t_i \right\|^2 \tag{2.3}$$

Since E is a function of the outputs and therefore also a function of the activations a, the weight update can be calculated as shown in Equation 2.4.

$$\frac{\partial E}{\partial w_{ij}^l} = \underbrace{\frac{\partial E}{\partial a_j^l}}_{\delta_j^l} \cdot \frac{\partial a_j^l}{\partial w_{ij}^l} = \delta_j^l \cdot \frac{\partial a_j^l}{\partial w_{ij}^l} = \delta_j^l \cdot o_i^{l-1} \tag{2.4}$$

The second part of this equation can be calculated directly whereas the first part has to be further investigated. This part can be seen as the backpropagated error and is usually referred to as δ. Again applying the chain rule yields a recursive definition of the values for δ as defined in *Equation 2.5.* The initial δ for the last layer where backpropagation is started is equivalent to the derivative of the error function. The definition of the δ includes the derivative of the activation function with regard to its inputs, thus requiring the activation function to be differentiable.

$$\begin{aligned}
\delta_j^l &= \frac{\partial E}{\partial a_j^l} = \sum_k \underbrace{\frac{\partial E}{\partial a_k^{l+1}}}_{\delta_k^{l+1}} \cdot \frac{\partial a_k^{l+1}}{\partial a_j^l} \\
&= \sum_k \delta_k^{l+1} \cdot \frac{\partial a_k^{l+1}}{\partial o_j^l} \cdot \frac{\partial o_j^l}{\partial a_j^l} \\
&= \sum_k \delta_k^{l+1} \cdot w_{jk}^{l+1} \cdot \frac{\partial f}{\partial a_j^l}
\end{aligned} \tag{2.5}$$

Since a simple threshold function is not differentiable, usually sigmoid functions are used as activation functions. Mostly the logistic function or the hyperbolic tangent are used. However, as will be shown in section 3.2 using rectified linear activation functions improve learning and generalization. Actually rectified activation functions are not differentiable at 0. In practice however this does not matter.

With the logistic function as activation function each layer in a neural network, exactly performs a logistic regression on its inputs. This means that each layer learns to classify its inputs by fitting a line (or hyperplane in higher dimensions) yielding a probability for each class. Each layer transforms the data, finally allowing the last layer to separate the desired classes.[59]

A simple example of this behavior is shown in Figure 2.4. Whereas a single layered network can only learn to separate the blue and red class by a straight line as shown on the left, a network with a hidden layer can first transform the data and then linearly separate the two classes. In the center of Figure 2.4 this separation is visualized in terms of the input space whereas on the right the same learned discrimination is shown after transformation by the hidden layer.

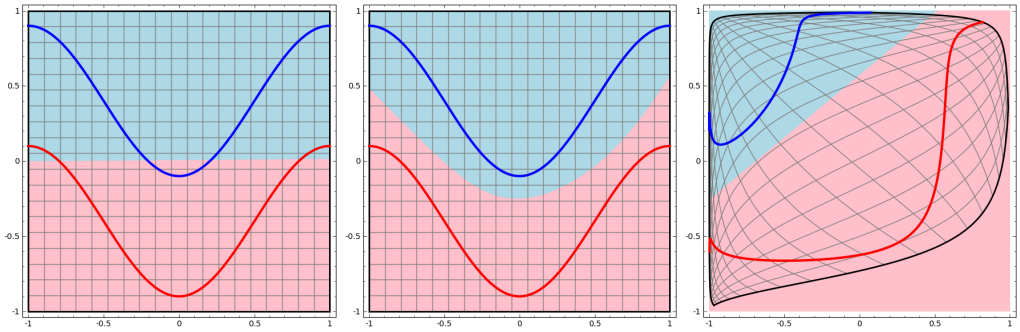

Figure 2.4: Classification example, left: possible solution of single layered neural network, center: solution of a neural network with one hidden layer, right: same as middle as seen by the output layer.[59]

2.5 Convolutional Neural Networks

The issue with using neural networks for image pattern recognition is the high dimensionality of the images. With fully-connected neural networks this would lead to a huge amount of trainable parameters. Furthermore fully-connected neural networks would not exploit the natural structure of images, like the high-correlation of neighboring pixels and repeating patterns.

The basic ideas of CNNs are first described in [24] by Fukushima where he derives a hierarchical neural network architecture based on the findings of Hubel and Wiesel for the visual cortex of cats and Macaque monkeys with alternating simple and complex cell layers and shows by simulating this artificial neural network that it is able to learn to recognize digits and letters in an unsupervised manner.[34, 35, 36]

Later, Homma et al. independently proposed the usage of the convolution operation and transfer functions (filters) instead of a full-connection scheme for extracting features from audio signals. Based on these findings Lecun et al. extended the ideas

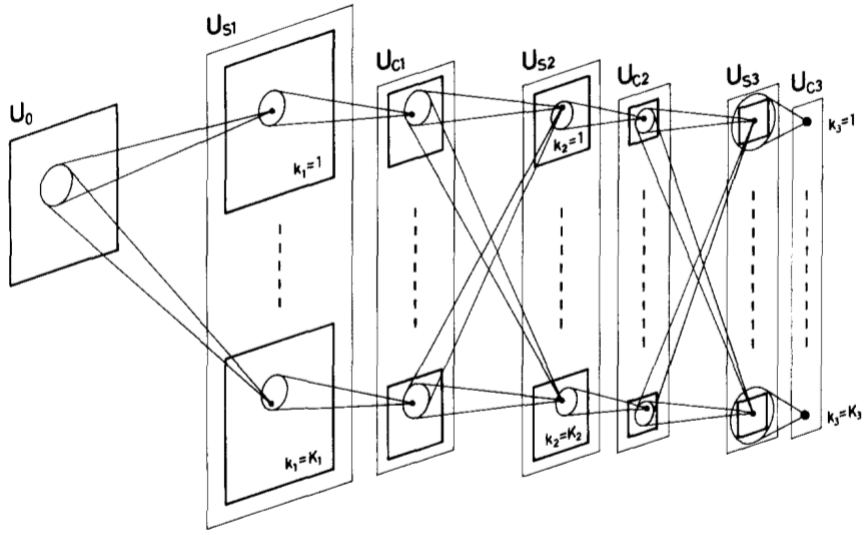

Figure 2.5: Neocognitron with alternating simple (U_S) and complex (U_C) cell layers. [24]

in [48] and established the term convolutional neural network. They showed that CNNs can be trained with backpropagation in the same way fully-connected neural networks were trained before and stated more precisely the different layer types by using the terms convolutional and pooling layer instead of simple and complex cell layer.

They prove the powerfulness of the general idea of alternating convolutional and pooling layers by supervised training of the LeNet5 network, which is shown in Figure 2.6, on handwritten digits performing end to end training, which is made possible by using a flat neural network as classifier after the convolutional network part.

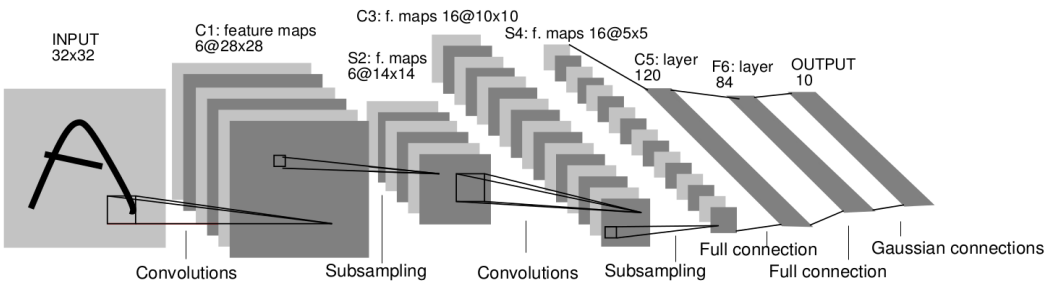

Figure 2.6: LeNet5 [48]

2.5.1 Convolution

The mathematical convolution operation that gives CNNs their name is defined as the integral product of two functions whereas one of them is considered the kernel or filter that is applied to the other function as defined in Equation 2.6. The kernel function however is reversed and shifted when applied. The result of the convolution is the weighted average of the two functions and can be interpreted as a measure of similarity, which makes it a good candidate for feature extraction. In the two-dimensional discrete form (defined in Equation 2.7) it is equivalent to cross-correlation with a kernel rotated about 180°.

$$f(x) \star g(x) = \int_{\alpha=-\infty}^{\infty} f(\alpha)\, g(x-\alpha)\, d\alpha \tag{2.6}$$

$$v(x,y) = \sum_{x_k=0}^{N_k-1} \sum_{y_k=0}^{N_k-1} i(x-x_k, y-y_k)\, k(x_k, y_k) \tag{2.7}$$

The convolution is an integral transform and thus a linear operator that transforms data from one domain to another domain.[20] This allows a CNN similarly to MLPs to learn to transform data using visual features into domains where separation can be achieved by a hyperplane.

2.5.2 Pooling

The rationale behind pooling layers is to reduce the size of the feature maps produced by the convolutional layers gradually by considering only the features with the highest response in a window of predefined size. This is the case for maximum pooling layers, however also average pooling have been used, where the average of the response was used instead of the maximum. However, average pooling was mainly utilized in the early years of CNNs whereas today maximum pooling is employed almost exclusively.

Another advantage of using pooling layers is, that they introduce translation invariance, i.e. the activation for a feature at different positions in the input space can have the same results which increases generalization. This makes maximum pooling superior to other pooling operations.[68]

2.5.3 Regularization

The basic concept of alternating convolutional and pooling layers did not change in recent years. However there is a trend to deeper, i.e. networks with more convolutional layers, and wider, i.e. more neurons in a layer, architectures as shown in [46, 71] describing the 2012 and 2014 winning CNNs of ILSVRC. This is made possible by advances in the availability of computational resources on the one hand but also improved training algorithms and improvements regarding the network structure on the other hand.

Deep network architectures with their high number of trainable parameters make training hard not only because of the computational resources required, though. Overfitting is a major problem because of the great complexity of the model compared to the already big but still too small amounts of data. Consequently much work was done addressing this problem, either by artificially increasing the amount of training data or improving the structure of neural networks to increase the inherent ability to generalize.

Data augmentation is one way of increasing the dataset size. It uses image transformations that do not change the label of an image. The simplest forms of augmentation are horizontal mirroring and random cropping. In [46] each input image of size 256×256 is randomly cropped using a size of 224×224. Cropping and mirroring increases the dataset size by a factor of 2048 and therefore reduced overfitting significantly. Another mechanism applied in [46] to further increase the dataset size uses the principle components of the original dataset to randomly transform an image's RGB intensities.

The choice of non-linearity for the neurons and applied normalization also effect the models accuracy significantly, as reported by Jarrett et al. in [40]. Although rectified activation functions were already used in [24], mostly sigmoid activation functions have been used later. After the work of Jarrett et al. on the influence of the choice of activation function, rectified linear units (ReLUs) as defined in Equation 2.8 have been proven to have positive effects on training speed and generalization as reported in [58] for restricted Boltzmann machines and in [46] for training deep CNNs.

$$r(x) = \max(0, x) \tag{2.8}$$

For training CNNs, this is due to ReLUs not saturating in contrast to sigmoid activation functions which speeds up gradient descent very much.[46] Also in [40] the authors reckon that ReLUs improve performance because the polarity of a feature is mostly unimportant for object recognition. ReLUs can thus be seen as one of the building blocks that enabled the training of more complex networks.

One important structural method to improve generalization is *dropout*. As described by Srivastava et al. in [70], *dropout* simulates an ensemble of networks by randomly

dropping units and their connections from the network during training. That means, considering the set of networks that contains all copies of the original network with an arbitrary number of neurons removed, *dropout* is equivalent to training this whole set whereas all networks share the same weights. One network is randomly sampled from this set for each training iteration for both the forward and backward pass. Thus only a subset of neurons of the original network is altered in each iteration. After finishing training the net as a whole is used for prediction, but with weights down-scaled proportional to the *dropout* probability, which approximates the averaging of the predictions of all models.

As a result *dropout* can be seen as adding random noise to the neurons, preventing them to respond too much to noise in the dataset and improving generalization of the whole network.

3 State of the Art

This chapter gives an overview of related work in the field of wildfire smoke detection as well as image and video recognition using convolutional neural networks in general.

3.1 Smoke Detection

There are many publications on wildfire smoke detection, however comparing the different approaches is very hard, because there are no wildfire smoke datasets with reasonable size and quality. As a consequence publication results on smoke detection remain vague or questionable, since mostly only a small number of images or sequences is used for evaluation.

The few available datasets and the most common approaches in the field of smoke detection are described in this section. Due to the lack of data, the effectiveness of the presented approaches can not be assessed, though.

3.1.1 Datasets

The university of Bilkent released a number of videos of different length that contain smoke and fire along with their smoke and fire detection software *VisiFire*.[1] All in all there are 33 videos of small controlled fires, real wild fires and videos without smoke. The camera view angle and position varies very much. Labels are given globally for a whole video only. Some videos contain more than a single smoke source and negative examples are videos of traffic taken at night. The resolution ranges from 320×240 to 720×576 pixels at a frame rate of 10 to 30 frames per second.

The center for wildfire research of the university of Split runs a website that aims at collecting real wildfire smoke image and video data.[2] They provide two still image datasets. Both contain smoke images taken from the ground and from the air. While the first contains an unknown number (data can be accessed after registration only) of unlabeled still images, all containing smoke, but without labels, the second contains 98 segmented images. A video dataset with six videos is also available. Five sequences contain smoke and one does not contain smoke. The ground truth is provided as a mask for each frame. Another video dataset contains ten videos

with a resolution of 1200 × 676 at 25 frames per second. All videos use stationary cameras.

Due to the lack of data Labati et al. in [47] propose to create smoke image sequences by generating smoke plume that is inserted into an image sequence without smoke. However, the resulting images appear to be not very realistic.

3.1.2 Motion Detection

Most of the proposed models leverage temporal information thus working with image sequences. Background estimation is utilized in almost all publications, e.g. [62, 18], to detect moving objects.

One simple, yet effective algorithm that is widely used was proposed in [19] by Collins et al., which combines frame differencing and adaptive background subtraction. This algorithm models a pixel-wise threshold for determining moving pixels regarding the background that is estimated in parallel. Starting with the first image in the sequence as the initial background estimation and some fixed thresholds, subsequent frames are background subtracted and pixels are considered moving when above the current threshold. Background and thresholds are then updated for stationary pixels only taking the new information into account.

Another approach to extract temporal information is to calculate the optical flow between subsequent images. The optical flow field describes the velocity vectors for each pixel considering the relative motion between an observer and a scene. Estimating optical flow is a quite hard problem, since assumptions have to be made to calculate the optical flow at all. E.g commonly pixel intensity values are assumed not to change due to motion, which is of course violated in many cases. Furthermore presuming changes to be local leads to the so called aperture problem, so that the motion can not be uniquely resolved.[23]

As a result a variety of optical flow estimation methods exist. One of the most commonly used methods is the Lucas-Kanade method as described in [52]. This differential method uses least squares to estimate the optical flow constrained to a local neighborhood. Smoke detection algorithms that rely on optical flow features are described for example in [18, 15, 62, 8, 49].

3.1.3 Features of Visual Nature

Another widely used set of features relates to the visual appearance of smoke, i.e. color, shape, color gradients and chromatic features. For example [8, 47, 62] use these features to further refine image regions extracted after motion detection, allowing to distinct smoke from other moving objects like cars or trees that move in the wind.

3.1.4 Dynamics Modeling

Modeling the inherent dynamics of the smoke plume can further help discriminate smoke from other objects. The dynamics models usually build on top of the motion detection that was already described. The convection produced by the fire's heat produces a particular motion of the smoke from the inside to the outside in addition to motion induced by wind which can be tried to be detect.

In [8] Green's theorem (which is equivalent to the two-dimensional divergence theorem) is adopted to test potential smoke sources regarding their divergence, i.e. their emissive character. By first determining the optical flow field, they then test a rectangular region around a smoke source candidate to calculate the outward flux for this region which is equal to the divergence of the region.[3]

Other dynamics used as features involve the average upwards motion above the source also used in [8], the variation and average of the optical flow velocity and the variation in orientation as utilized by [18].

3.1.5 Smoke Detection using CNNs

At the time of writing of this thesis there have been no publications on wildfire smoke detection using CNNs. However, in the rather old paper [72] from 2006 Tivive and Bouzerdoum do texture classification with a CNN which can be considered related to smoke detection. They train a CNN with shunting neurons rather than with neurons with a sigmoid activation function. The task of the network is to classify single pixels regarding the texture. The input to the network is a 13×13 patch whose center pixel is going to be classified. The network architecture is quite small both in the number of layers and the free parameters. Also noteworthy is that Tivive and Bouzerdoum propose to use a sparse binary connection scheme between the convolutional layers instead of fully connecting subsequent layers which greatly reduces the complexity of the network. Experiments with 2, 5 and 10 classes are made training on data taken from [61]. Unfortunately, it is not clear how many images were used for training. They conclude that their CNN performs very well on texture classification in comparison to approaches that utilize wavelets or quadrature mirror filters.

3.1.6 Fshell

The *fshell* algorithm was developed by the DLR in 1997 and patented, so that a description of its functionality is available in [10]. It uses four major steps to determine whether there is smoke in an image sequence. None of the used filters applied by the algorithm was learned but created by sophisticated trial and error and much experience with visual recognition tasks in general. The first image in the sequence is used as reference image for all subsequent images.

In the first step each image is matched with the reference image to remove small movement of the camera. Furthermore, the horizon is identified and difference images are created. Estimating the horizon allows on the one hand to discard most of the area above it and on the other hand to create a distance map along the y-axis of the image using the tower height. Based on the distance information minimum sizes for the smoke plume can be defined.

Next, the algorithm uses thresholding to create binary images from the difference images and determines connected regions in these black and white images. The rationale for this is to find regions of moving objects.

Third, the extracted regions are rated according to their correlation through the whole sequence using the original images. It is assumed that regions of smoke are uncorrelated, since the smoke's appearance is constantly changing whereas other moving objects like cars, planes or people do not change their appearance. Additionally, the region's contrast is used to further exclude non-smoke regions.

Finally, a fractal feature is applied that accounts for the specific visual structure of smoke. It measures the slope and y-intercept of a line fitted to the relation between the pixels logarithmic absolute difference values and their logarithmic distance.

Based on the application of the features each region is rated according to the probability for containing smoke. When at least one region's probability lies above a certain threshold, an alarm is raised.

3.2 Convolutional Neural Networks

CNNs experienced growing attention in the last few years because of their impressive results on image recognition tasks. This development is well documented by the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The challenge is a benchmark for state-of-the-art algorithms in three different image recognition tasks. These are: image classification, single-object localization and object detection. In the early years of ILSVRC mostly algorithms with handcrafted features dominated the challenge. When the SuperVision team won 2012 with their very deep CNN in the categories image classification and single-object localization, it marked a turning point in image recognition, because from this point until now all of the top competitors use CNNs for all three tasks.[67] When considering the current state of research in the area of CNNs, the ILSVRC is thus a very good point of reference.

3.2.1 Architectural Advances

Different approaches have been made to optimize the architecture of CNNs. Goodfellow et al. created the concept of *maxout* units, that instead of using a fixed

activation function are able to learn it. The idea was to use a small single layered neural network as activation function that is able to linearly approximate any convex functions, with the advantage that *dropout* can be utilized accurately.

Based on the idea of *maxout* units Lin et al. took it a step further, suggesting to use multi-layered neural networks as activation function, arguing that filters of CNNs are generalized linear models and thus their level of abstraction is low, but extracted features should be invariant to instances of the same concept and thus a universal function approximator should be used. The idea of Lin et al. was consequently labeled *network in network* (NIN).

Inspired by the NIN architecture and theoretical work by Arora et al. Szegedy et al. derived in [71] an architecture that they named *Inception*. Addressing the two major issues with deeper and wider network architectures, increased demand of computational resources and large amounts of data, they try to find an architecture that mimics sparsely connected layers in a fashion that is efficient for the current hardware.

Sparse networks have the advantage of less learnable parameter and lower complexity, however modern hardware does not benefit from sparse matrices when it comes to matrix multiplication, since it is highly optimized for dense matrix multiplication.

The desire for sparse networks also comes from the findings in [9] by Arora et al. They prove that a randomly generated very sparse deep network can be resembled by a layer by layer constructed network that is optimal regarding its topology by basically following the Hebbian rule that is casually summarized by the statement of Carla Shatz: "Cells that fire together, wire together". This means that when the activations of units in a layer correlate, they should be connected to the same unit in the next layer. Their findings can also be described as follows: If for a given dataset probability distribution a very sparse deep network exists, the optimal architecture can be created layer by layer following the aforementioned rule.

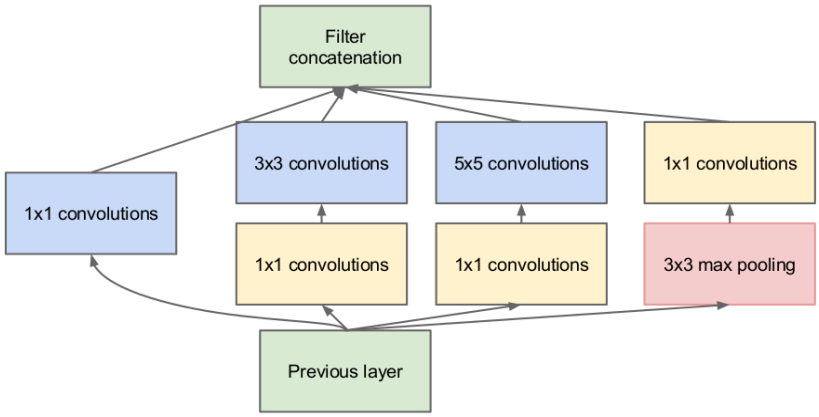

Figure 3.1: A single inception module[71]

The architecture Szegedy et al. developed consists of so called inception modules as shown in Figure 3.1. A single module consists of convolutions with different filter sizes. The idea behind this is, that correlations in feature maps can appear in a local neighborhood but at different scales. The yellow 1×1 convolutions could be omitted. However, 3×3 and 5×5 convolutions are very expensive, so that these 1×1 convolutions were introduced to reduce the number of feature maps and thus the dimensionality before convolving with the bigger filters. Pooling is also part of an inception module, because it has proven to be beneficial for training CNNs.

All convolutions use ReLUs. The resulting feature maps are concatenated to form the output of the inception module. This is possible due to the utilization of padding for all convolutions.

The whole network architecture is then made of several inception modules stacked upon each other. However, Szegedy et al. state that using inception modules at lower layers was omitted due to efficiency reasons and instead regular convolutions were used. Following [50], they also use average pooling instead of fully-connected layers for the classification part.

The authors showed the powerfulness of this architecture by winning the ILSVRC14 competition. The used *GoogLeNet* consisted of nine inception modules totaling to 22 layers with learnable parameters.

3.2.2 Object Detection using CNNs

CNNs have proven to give very good performance when applied on image classification tasks. However, extending the use of CNNs to object detection and object localization is not straight forward. A CNN is always trained on a fixed number of classes, so when applied to object localization it is necessary to introduce a "background" class that contains anything that does not fall in any of the classes to be detected. The naïve way to do object localization with a CNN is using a sliding window approach which is very computationally intensive.

A more sophisticated and very relevant approach in this field is using region proposals as suggested by Girshick et al. in [28]. They show that using selective search to extract region proposals that are then fed into a CNN improves the mean average precision significantly compared with competing models on the PASCAL VOC and ILSVRC2013 challenges. Since the method combines region proposals and CNNs, it is labeled R-CNN (not to be confused with recurrent CNNs, however).

Instead of fully-connected layers Girshick et al. utilize one support vector machine (SVM) for each class. Thus the CNN serves as a feature extractor only and each SVM uses this output to predict the class. However, they state that fully-connected layers for classification will probably yield comparable results.

They also compare different mechanism for transforming the proposed regions to fit the input of the CNN. The simplest mechanism is warping the proposed region to

the CNN input size. Other approaches are "tightest square with/without context" where a square is fitted around the region and additional image data that falls in this square is included or excluded respectively. Furthermore context padding with various sizes was tested, i.e. including a number of pixels around the original region proposal. From these approaches warping with a margin of 16 pixels worked best for an input of 227×227 pixels to the CNN.

Some general findings by Girshick et al. regarding CNNs are quite insightful as well. Confronted with data scarcity, they try using pre-trained models and then fine-tune it with a smaller dataset for different classification tasks. Their results show that this approach yields substantial improvements compared with training from scratch, which means that filters learned by a CNN generalize very well.

Additionally, they perform an evaluation of the last three layers of the employed CNN regarding the importance for the detection performance. These layers are the last pooling layer and two fully-connected layers. As already stated the fully-connected layers were not used for classification, but to train the whole CNN these layers are mandatory and can therefore be evaluated.

When evaluating these layers for the VOC 2007 dataset prior to fine-tuning, they find that the features extracted by the last fully-connected layer generalize worse than those from the layer before. Removing both fully-connected layers leaving the CNN with only 6% of the actual parameters, still gives good results. After fine-tuning, the improvement for the fully-connected layers is much higher than for the pooling layer. This lets the authors conclude that most of the capabilities for abstraction of the CNN comes from the convolutional part whereas the fully-connected layers are very domain specific.

He et al. in [31] raise the question why CNNs require a fixed input size, since this puts an artificial restriction on object detection with CNNs. Clearly, the convolutional part of a CNN can be easily applied on inputs with different sizes, but the fully-connected classifier part requires a fixed input size, as the authors find. Therefore they suggest introducing a spatial pyramid pooling layer that is added between the convolutional part and the fully-connected part and turns the output of the first, that varies in size, into a fixed size feature vector that can be fed to the first fully-connected layer.

Because He et al. can process variable sized inputs with their architecture, they further elaborate that this is advantageous for object detection, too. Instead of extracting a number of region proposal prior to processing by the CNN, the whole image can be used as input. From the calculated feature maps they then extract the features for the region proposals and predict the class for each proposal. This approach is much more efficient when the number of region proposal is large, because features are extracted only once for the whole image. However, training such an architecture is more difficult, because current frameworks for training CNNs require a fixed input size.

3.2.3 Spatio-temporal Feature Learning

CNNs were initially developed for two-dimensional data as images and audio signals. However, they can be extended to exploit data with higher dimensions, such as videos, as well by extending the two-dimensional convolutional filters to a third temporal dimension. These CNN models will be referred to as C3D networks.

Increasing the kernel dimension of a CNN, leads to filters forming a cube rather than a square. Filters are no longer learned exclusively regarding nearby pixels in a single image but also along pixels that are close in time, i.e. in subsequent images, thus learning spatio-temporal features. This differs from just stacking video frames and using these as input feature maps to a two-dimensional CNN as this approach does not preserve temporal information and thus is unable to model motion.[73]

C3D networks have been applied on various video recognition tasks recently mostly comparing the performance to single frame CNN approaches.

In [41] a C3D network is trained on the TRECVID dataset to recognize three classes of human actions based on image data taken from surveillance cameras at London Gatwick airport. Ji et al. developed an architecture shown in Figure 3.2 where the first convolutional layer utilizes hardwired kernels to produce 33 feature maps in five channels. In addition to a grayscale version of the input data, these five channels already extract motion information such as the gradient and the optical flow fields horizontally and vertically. In the following convolutional layers a 3D convolution is applied separately on each of the channels using only a small number of feature maps per channel, i.e. two in the first convolutional layer, six in the second and third. Only the first two convolutional layers run over the temporal dimension using a kernel size of three, whereas pooling is done along the spatial dimension only.

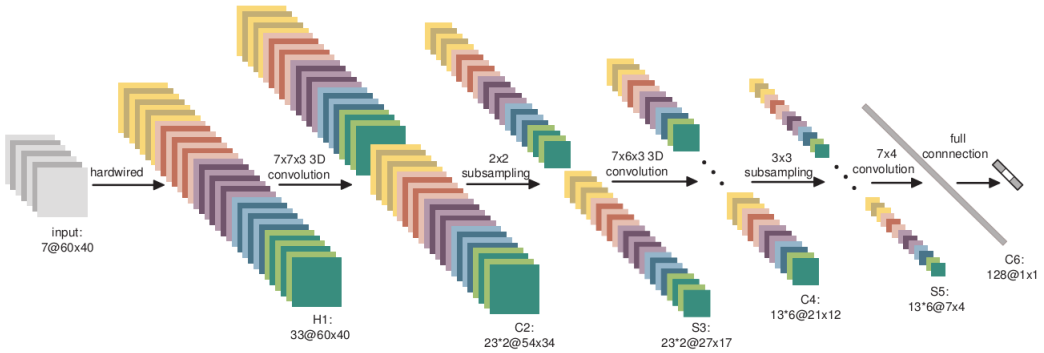

Figure 3.2: C3D network architecture from [41].

Despite the quite small network architecture, the authors show that their C3D network outperforms frame-based CNNs and state of the art algorithms for human action recognition. Furthermore they find that C3D networks perform better when the number of positive examples is small, which seems rather surprising.

A clearer view on C3D architectures provide Tran et al. in [73]. They directly train different C3D networks on subsequent video frames where all filters are learned. They compare network architectures where the convolutional layer's kernel size is only varied along the temporal dimension. Additionally, they include pooling along the temporal dimension except for the first pooling layer to preserve the temporal information for higher layers. The number of feature maps is comparable to two-dimensional CNN architectures. The networks are trained on the UCF101 dataset that contains action videos from YouTube in 101 different classes.

Furthermore, a C3D network using a similar architecture as the *VGGNetA* only extending the kernel dimension was trained on the I380K dataset that contains 380k Instagram action videos and 382 classes, which yields better results compared to *AlexNet* and the two-dimensional *VGGNetA* with an improvement of 4.5% and 2% respectively.

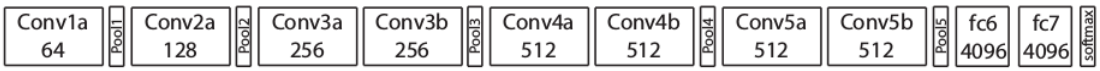

Figure 3.3: C3D network architecture from [73], all convolutional kernels have size 3x3x3 and are applied with a stride of 1 and padding.

They find that C3D networks are more suitable for spatio-temporal image data than two-dimensional CNNs. Regarding different architectures they state that small kernel sizes — specifically a size of three for the temporal dimension — and deeper networks perform better. The network architecture used in this paper is shown in Figure 3.3.

Similarly, in [44] Karpathy et al. conclude that C3D networks perform significantly better compared to strong feature based models. On the other hand, they only find minor improvements when compared with single frame two-dimensional CNNs. However, as the authors state themselves, this could be due to the used dataset (Sports-1M) which was made of one million YouTube videos with 487 classes (1000-3000 examples per class) and weakly annotated using an automated algorithm that used the video's description and comments to label the video.

Most important, the camera is not static in some videos and the authors come to the result that camera motion decreases the performance of a C3D network especially when compared with single frame CNNs. Also in videos where motion does not play an important role, single-frame CNNs performed better.

Regarding the network architecture the paper investigates different points in the network when to use the temporal information, which they call *time information fusion*.

The temporal information can be fused early, late or slow as can be seen in Figure 3.4. In the case of late fusion the network consists of two single-frame CNNs where one receives the first and the other the last frame. Only spatial features are extracted

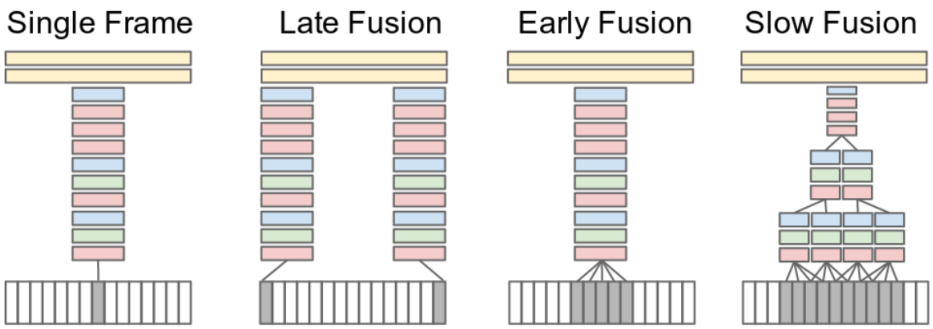

Figure 3.4: Time information fusion, layers are color coded: Convolutional – red, normalization – green, pooling – blue [44]

until both subnetworks end in a fully-connected layer that then can use temporal information based on the differences produced by the two CNNs.

The other extreme is early fusion where the first convolutional layer uses filters with a great temporal dimension, thus consuming all time information.

Finally, slow fusion is the compromise between these two utilizing convolutional layers with smaller filters along the temporal dimension, so that higher layers have access to more temporal information. Comparing these different network architectures the authors come to the conclusion that slow fusion performs best.

4 Wildfire Smoke Data

The data that was used for training and validation and the definition of ground truth is described in this chapter. Further, the size and aspect ratio of the smoke is analyzed based on the ground truth data.

4.1 FireWatch Data

The DLR receives image sequences and fire alerts from the *FireWatch* system which is deployed all over the world. Thus data from different locations was available. However, data from different countries varies a lot in quality and quantity. Besides different camera models being used with different resolutions, vegetation and landscape is very different too. Since a great fraction of the data collected originates from Germany and this data has the highest quality regarding information about the presence of a real wildfire in the images, the decision was made to rely on this data for training and validation.

As a result all tagged image sequences containing smoke from Germany till the 3rd of July 2015 were collected to form the data pool for training and validation datasets. This base dataset for training and validation contains 1,835 sequences with different length resulting in 5,980 images. I will refer to this dataset as *Germany-1835*.

Generally, the data is organized in image sequences of varying length but with at least three and at most eight images. The images have a size of 1024×512. A sequence is associated with a tower, direction and timestamp forming a unique identifier or name of the sequence.

4.2 Ground Truth

As already mentioned all sequences in the dataset contain smoke, but the parts in the images containing smoke are mostly very small compared to the overall image size. As can be seen in Figure 4.2, half the ground truth rectangles have a height smaller than 70 pixels. The ground truth was created by different people by defining a single rectangle that encloses the visible smoke in all images of the sequence. This single rectangle is the binary ground truth. Thus it is mostly a rather rough approximation to the real smoke's shape an the smoke may cover only a small fraction of the ground

truth rectangle. This is also due to the fact that the ground truth rectangle is defined for the whole sequence and does not change. So when the smoke is small in the first but large in the last image, the ground truth rectangle is defined so that it encloses the large smoke plume.

From my own experience I can tell that the labeling of sequences can be very difficult. Especially specifying where the smoke ends can be tough and depends not only on the sequence at hand but also on the quality of the monitor used. As a result the quality of the ground truth varies quite much. An example of image patches defined by a sequence's ground truth can be seen in Figure 4.1.

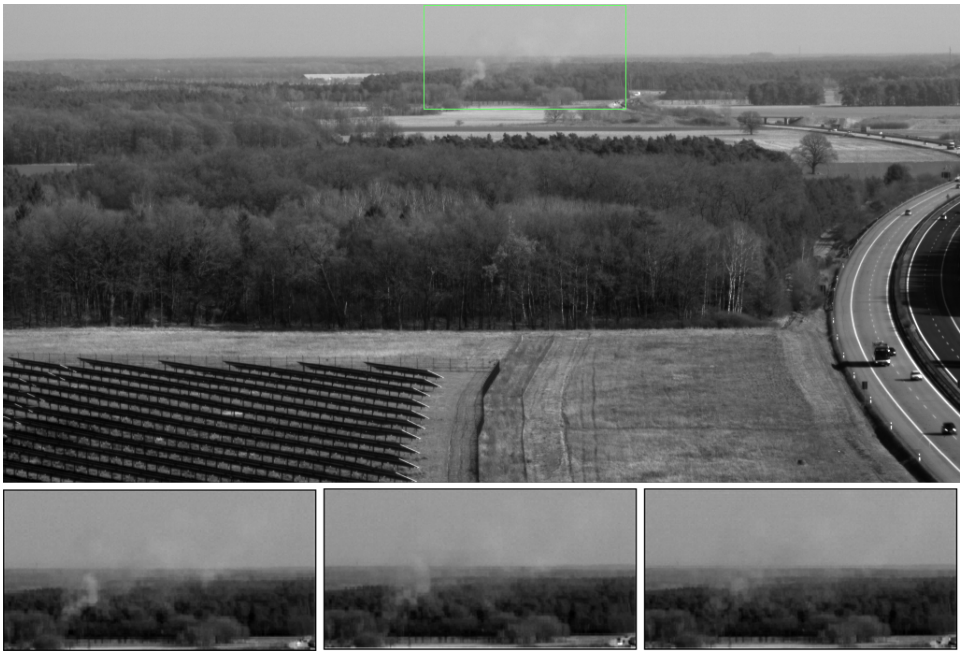

Figure 4.1: Ground truth example, top: first full size image in the sequence, bottom: extracted ground truth patches from the whole sequence.

Since only ground truth rectangles were available, pixel-wise classification, i.e. image segmentation, was impossible. Further ground truth like the smoke's base point was also not available.

CNNs have a fixed input size and it therefore makes sense to investigate how the ground truth's shape is distributed to decide which input size to use. A smaller input size reduces the number of learnable parameters and would make training faster. However choosing the input size too small, would require down-scaling image segments a lot discarding much information that might be necessary to learn meaningful features.

Figure 4.2 depicts the distribution of the ground truth's width, height and aspect ratio as histograms for the whole dataset. The aspect ratio has its maximum around 1.5 and its median at 2, which suggests that the ground truth generally has a greater width than height.

Although width and height are two independent random variables, their histograms can be used as indication to properly select the height of the resolution. The charts suggest that using a height of 100 pixels and a width of 200 pixels seems to be reasonable taking also into account the aspect ratio's median of 2.

When using the image patch defined by the ground truth as positive example, there is the issue of how to fit the patch size to the models input size. Different approaches are discussed in the literature.

Girshick et al. consider warping an image patch to the desired size, i.e. resizing the patch without considering its aspect ratio so that the result is distorted. They find that this method worked well, however in the case of smoke this led to very poor results.

Another solution to the size fitting issue is to fit a rectangle with the model's input aspect ratio around the region defined by the ground truth. This approach can be problematic for extreme aspect ratios.

As can be seen in Figure 4.2 there are some ground truths with great width and small height. Fitting a rectangle in this case leads to an image patch that is much larger than the original region defined by the ground truth so that the part containing smoke in this patch gets very small.

Another approach was therefore considered that tiles the rectangle defined by the ground truth. A rectangle with the input's aspect ratio is used to slide over the ground truth. This rectangle slides either horizontal or vertical depending on whether the input's aspect ratio is smaller or greater than the ground truth's aspect ratio. The step size for sliding the rectangle was chosen to be half the width or height of the rectangle for horizontal or vertical sliding respectively. Tiling the ground truth is similar to cropping and can be considered a type of data augmentation thus increasing the amount of positive examples that can be used for training. Finally, a combination of the fitting and tiling policies can be used, too.

4.3 Germany-1835 Training and Validation Sets

In machine learning it is common practice to define three different data sets: training, validation and test set. The first is used to train a model, i.e. trying to find statistical patterns that can be used to discriminate different classes in the case of classification. It is important though, not to overfit the training data, which means that the model learns patterns that do not generalize but hold for the specific training set only. To determine, during training, whether a model is overfitting the data, the validation set is used. It furthermore can be utilized to measure a model's performance allowing to compare different models and tune hyper-parameters. Lastly the testing set is employed to ascertain how well the problem is solved by the finally chosen algorithm. It is important that these three datasets are not mixed, otherwise the results for the validation and test sets can be misleading.

Different types of training and validation datasets are distinguished for the *Germany-1835* dataset. First, the sequences in this dataset have to be divided to be either used for training or validation. These datasets will be called validation sequence and training sequence dataset. In a next step, region datasets will be derived from these sequence datasets that can be used to train and validate a model. According to the training and validation sequence datasets there are training and validation region datasets. This distinction acknowledges the two parts of the smoke detection problem described above.

In this thesis, 1,835 image sequences that were recorded on 100 different towers located in Germany have been used for training and validation. These sequences had to be divided to form the respective sequence datasets. One option to split the *Germany-1835* dataset is randomly choosing sequences for each set. However, sequences that are captured from the same tower are strongly correlated, so that the performance of a trained model could appear much better than it really is. As a consequence the set of sequences was split according to tower IDs. 20 towers were chosen for the validation sequence set leading to 310 (16.9%) sequences. The rest 80 towers with 1,525 (83.1%) sequences form the training sequence set as can be seen in Figure 4.3.

The number of sequences associated with a certain tower varies ranging from towers with only one sequence to towers with 127 sequences. The towers for the validation sequence set therefore should be carefully selected, so that there is a reasonable amount of variance in the data of both sets. This can be seen in Figure 4.4 which shows the partitioning of the training and validation sequence sets regarding the towers.

The training and validation sequence sets described here were fixed for all experiments. Anyhow, because the models are not trained on the full size images, the specific training and validation region sets for each experiment vary depending on which method was used to sample the images, i.e. the algorithm to create region proposals. Therefore these datasets are described in section 5.3.

4.4 Further Validation Sets

Since in this thesis no final model is selected for the problem of wildfire smoke detection, no separate test set was employed. Instead another dataset, that can be considered an additional validation set, has been used to evaluate the trained models.

This dataset contains only negative examples where the *fshell* algorithm raised many false alarms. The sequences from this datasets were all captured from a single tower that resides in Löcknitz a village in the state Brandenburg. This tower is not part of the *Germany-1835* dataset. I will refer to this dataset as *Loecknitz-Negative*.

This dataset is divided according to three different tower directions. These can be evaluated separately to get a better inside about what might trigger false alarms.

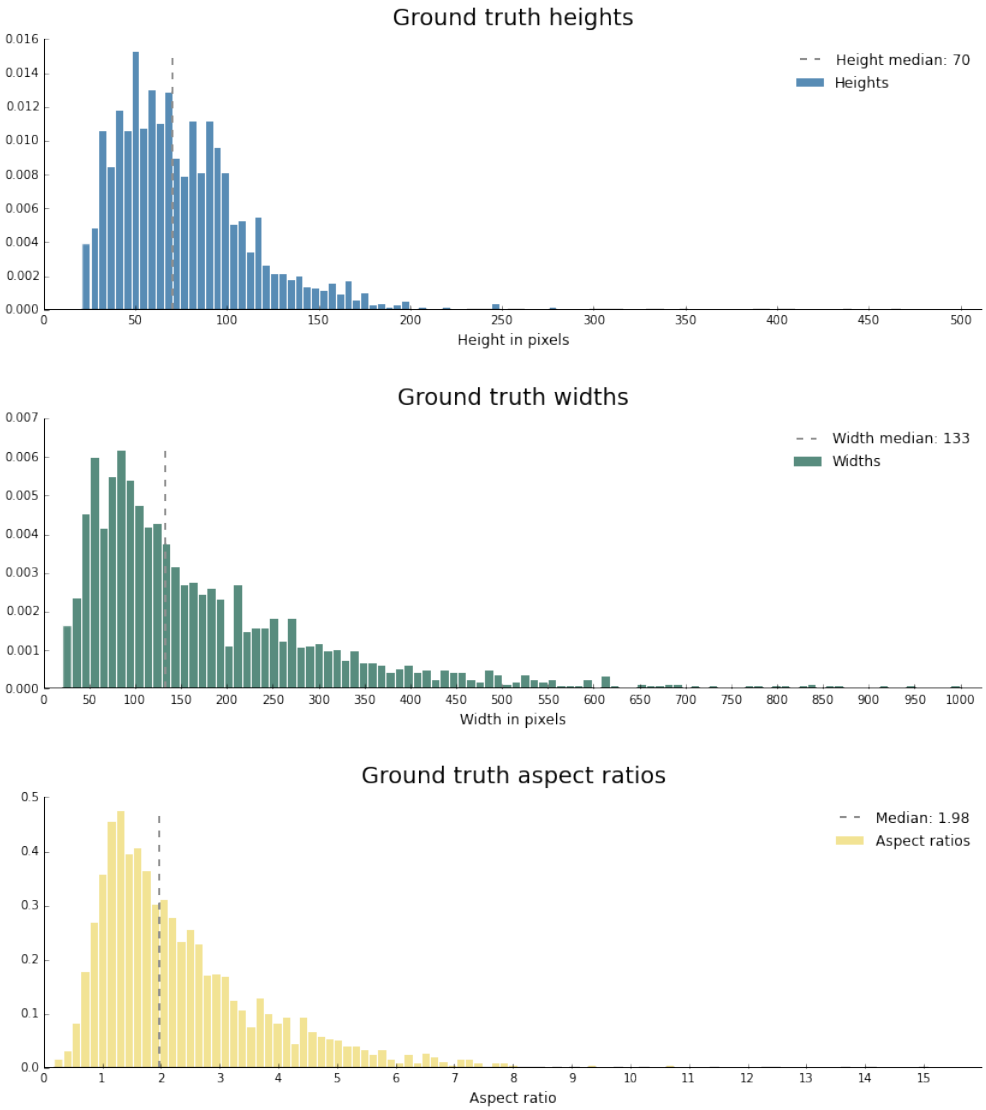

Figure 4.2: Distribution of the ground truth's width, height and aspect ratio for the selected dataset.

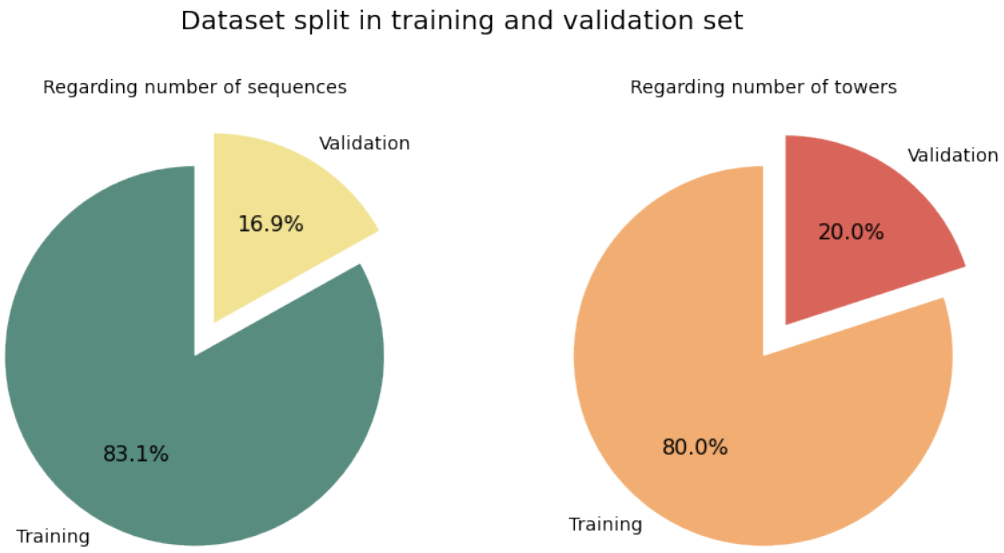

Figure 4.3: Fractions of the training and validation sequence sets, regarding number of sequences (left) and number of towers (right).

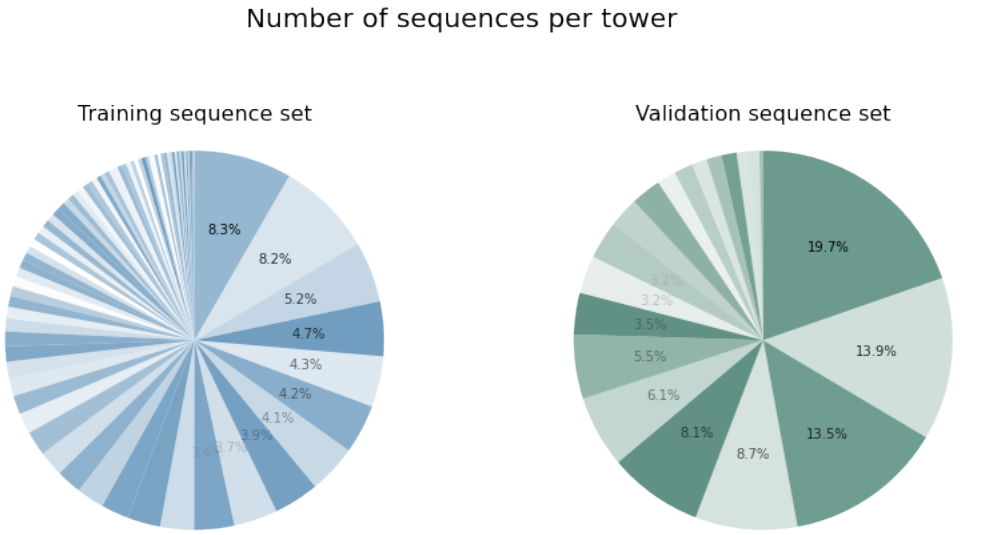

Figure 4.4: Fragmentation of training and validation sequence datasets with regard to the number of sequences per tower.

5 Methodology

In this chapter I will go into the methodological details regarding the setup of the conducted experiments. This includes the selection of a deep learning framework and the way data is preprocessed before training.

Moreover, this chapter describes the type of region proposal algorithm that was developed as an alternative to a sliding window approach. The choice of approach in this regard is an important factor for the performance of the CNN's. It can be seen as a weak classifier employed in a cascade with a CNN.

Finally, I will describe the models that where selected for training.

5.1 Deep Learning Frameworks

As the popularity of CNNs increased, different implementations arose. The most important factors of deep learning frameworks are their performance, usability, which includes the availability of APIs for different programming languages, quality of documentation and activity of the community. Performance is a very crucial requirement, since CNNs are complex models with many learnable parameters. Therefore CNNs are usually trained on graphics processing units (GPUs) instead of central processing units (CPUs). GPUs are hardware specifically optimized for executing many parallel matrix operations as required for graphics processing. CNNs have a inherently parallel structure and thus similarly benefit from a high level of parallelization.

Popular deep learning frameworks are *Torch*[4], *Theano*[12] and *Caffe*[42]. *Torch* is a framework for scientific computation that implements CNNs as well as other machine learning algorithms using the programming languages C/Lua where Lua is used as scripting language for using the framework. It is developed and used by Facebook AI, Google DeepMind and Twitter.

Caffe is a framework specialized for deep learning using C++ for implementation and Google protocol buffers for defining network architectures and controlling training. Furthermore, *Caffe* provides APIs for Python and Matlab. The development of *Caffe* is done by the Berkeley vision and learning center (BVLC). Both *Torch* and *Caffe* provide implementations for training CNNs on CPU and GPU.

Theano is quite different to *Torch* and *Caffe*. It is basically a Python library and thus also implemented in Python. Developers using the library write symbolic mathematical expressions that can then be executed on CPU or GPU. GPU execution

is implemented by automatically generating C code from these expression that can then run on a GPU. *Theano* can optimize the defined mathematical expression by deriving the given function. As a result *Theano* is not directly an implementation of CNNs only, but a more general library that can be used to easily implement CNN models. *Theano* was created by the Montreal institute for learning algorithms of the university of Montreal.

In this thesis *Caffe* was chosen as deep learning framework, because of its great performance, thorough documentation, ease of use and modularity which encourages developers providing implementations for state-of-the-art algorithmic improvements. Furthermore, *Caffe* introduced the so called Model Zoo. It on the one hand defines a format and on the other hand a place for sharing trained models, thus allowing researchers to build on top of each others work.

5.2 Preprocessing

Basic preprocessing was done at sequence level and single image level. First the images in each sequences were stabilized and then the contrast of each images was normalized.

5.2.1 Stabilization

Phase correlation was used to stabilize the sequences in a first step. This is required to enable the extraction of temporal features. Due to the cameras used to take the pictures being mounted on posts, there is sometimes shift between subsequent images resulting from the camera slightly moving in the wind. The method used for stabilizing each sequence is phase correlation, that is based on the Fourier shift theorem.

Phase correlation uses the discrete Fourier transform to determine the shift in phase of two images, thus working in the frequency domain of the images. Considering two signals $f(x)$ and $g(x)$ and their Fourier transforms $F(\omega)$ and $G(\omega)$, the normalized cross-power spectrum $R(\omega)$ can be calculated as shown in Equation 5.1 where $G^*(\omega)$ is the complex conjugate of $G(\omega)$ and $\circ$ denotes the entry-wise product (Hadamard product).[6]

$$R(\omega) = \frac{F(\omega) \circ G^*(\omega)}{|F(\omega) \circ G^*(\omega)|} \tag{5.1}$$

This is equal to the Fourier transform of the cross-correlation of both signals. Thus applying the inverse Fourier transform yields the cross-correlation of the signals,

which could be determined directly by cross-correlating the original signals. However, cross-correlating large images directly is very slow, compared to applying the fast Fourier transform and working in the frequency domain. Finally, finding the maximum of the cross-correlation of the signals yields the shift.

5.2.2 Contrast Normalization

After image stabilization each image's pixel intensities are rescaled to the range 0 through 1 with respect to the minimum and maximum pixel intensities of the whole sequence.

Training and validation happens on image patches extracted from the sequences. Therefore each patch's intensity values are also rescaled to the range 0 to 1, whereas when training is performed on a sequence of patches, the intensity is rescaled to the range 0 to 1, too but with regard to the sequence's overall minimum and maximum intensity values. Furthermore, the pixel-wise mean of all training patches is calculated and subtracted from each data point that is input to the CNN, so that the data is centered around the mean.

5.3 Region Proposals

When trying to detect only a single class of objects, the question arises what to use as negative examples. This is generally a problem when dealing with a binary classification and detection task. In the case of the wildfire smoke detection task, indeed every patch that does not contain smoke of an image from the distribution of available images could serve as a counterexample. Taking also into account different scales for these patches a huge number of negative examples exists.

Another point to consider is how a model is going to be applied to find smoke within a larger image after training.

Both, negative example generation and smoke localization in an image are interdependent, because the model's response to data presented at deployment depends on what the model has "seen" during training. In general this is the problem of region proposal. In this thesis two different approaches for region proposals have been subject to investigation.

5.3.1 Random Sampling

The first attempt of negative data generation was to randomly sample the whole distribution of patches not containing smoke with the goal to later apply a sliding window approach on a given image to decide whether smoke is present or not. As already described the distribution of negatives is very large in this case, so that a

Ground Truth Policy	#Positives	#Negatives			#Total
		1st Third	Rest	Total	
Training					
Fit	5495 (26.81%)	9000 (43.91%)	6000 (29.28%)	15000 (73.19%)	20495
Fit & Tile	29404 (66.22%)	9000 (20.27%)	6000 (13.51%)	15000 (33.78%)	44404
Validation					
Fit	1223 (28.96%)	2000 (47.36%)	1000 (23.68%)	3000 (71.04%)	4223
Fit & Tile	6878 (69.63%)	2000 (20.25%)	1000 (10.12%)	3000 (30.37%)	9878

Table 5.1: Training and validation region sets with randomly sampled negatives.

subset has to be selected. Assuming that the patches in the distribution are very redundant – a lot of the patches will show trees or treetops – taking random samples might resemble the class of non-smoke patches quite well.

Tests with random sampling showed that the trained models were very sensitive to the region around the horizon. The models seemed to have learned that the presence of the horizon is a strong indication for the presence of smoke. To counteract this, a greater number of negative examples were generated from the first third of the images. Also the sizes of the patches from this region were chosen to be smaller than in the rest of the image. This acknowledges, that searching for smoke in an image will also be done at smaller scale in the upper region of an image, because the smoke is smaller in this region due to the greater distance.

5.3.1.1 Germany-1835 Training and Validation Region Sets

Two different pairs of training and validation region sets have been considered for the experiments with random sampling. The first is biased towards the not-smoke class with a greater fraction of negative than positive examples. The ground truth policy applied in this case was fitting.

The second pair is biased towards the positive class. The same negative examples as in the first pair of datasets was used, but the positives were generated through fitting and tiling the ground truth. Table 5.1 lists the number of positives and negatives for both pairs in detail.

5.3.2 Selective Search

As will be exposed in chapter 6 the use of random samples for training yielded rather disappointing results. Although the performance of the trained models on the specific region datasets used for training and validation was very good, it was

very poor when scanning a larger image, which suggests that the random samples did not resemble the real distribution of patches. Thus selective search was considered as an alternative to exhaustively searching an image.

Selective search is a hierarchical partitioning algorithm specifically developed to suggest regions (rectangles) in an image that enclose objects, i.e. possible object locations, so that these regions could then be used to classify the objects found.[74] This kind of algorithm can be considered a weak classifier that is employed as the first stage in a cascade of classifiers, allowing classifiers at higher stages to be more computationally intensive, due to the reduced number of regions to be classified. This is in contrast to an exhaustive search with a single strong classifier.

The selective search algorithm was developed with the goal of a fast region proposal algorithm, yet yielding a complete set of object regions at different scales accounting for the hierarchical nature of images. Uijlings et al. use the image segmentation algorithm developed by Felzenszwalb and Huttenlocher to generate the initial set of region proposals, that are then greedily merged.

Felzenszwalb and Huttenlocher's approach to image segmentation uses a graph based model, where each pixel is considered a vertex and neighboring pixels are connected by edges, that are weighted according to the dissimilarity of the pixels. For gray-scale images this is the absolute difference in the pixel's intensity. Initially each pixel defines an image segment. Image segments are then greedily combined when the relative difference between them is smaller than defined by a threshold function that depends on the segment's size.

The algorithm's behavior can be controlled by two hyper-parameters. The first defines the threshold function used to determine whether two segments should be combined. Because the threshold function depends on the segment's size, this parameter effectively sets a preference for larger or smaller segments, where a small value supports smaller regions. The second parameter is the diameter of the Gaussian kernel used to smooth the image prior to running the algorithm.

Uijlings et al.'s selective search then calculates four complementary features to further merge the proposed regions. The first two features work in the color domain directly measuring the color similarity of two segments as well as the similarity in the texture based on SIFT-like features. The other two features are based on the segment's sizes. The first of which is the inverse of the relative size of the union of both segments with regard to the overall image size. Thus this feature produces higher values for smaller regions, so that they are merged early. The later is a measure of how well both segments fit together. It calculates the area not occupied by the segments in the tight bounding box around them with the goal to merge segments that are overlapping a lot. The combined measure of similarity is the sum of all four features.

With the set of initial segments generated through Felzenszwalb and Huttenlocher's algorithm, each pair of neighboring segments is rated regarding their similarity. The segments with the highest similarity are then merged into a new segment. The new

segment is added and its similarity to all remaining segments is determined. This procedure is repeated until there is only a single segment that encloses the whole image.

One major issue with smoke detection is that in contrast to other objects like for example cars, trees and houses smoke has no clear boundaries due to its semi-transparent fading nature. That is why selective search applied to single images performed poorly. The proposed regions in this case did capture the smoke in only 24% of the sequences. Thus two alternative approaches have been investigated that instead of applying selective search on the stationary single images, try to exploit the temporal information.

5.3.2.1 Selective Search on Difference Images

A simple way of discarding stationary information is to create difference images. As each sequence was stabilized, the remaining information that results from subtracting subsequent images reflects temporal changes in the sequence. As a result only a small fraction of the full images information remains. The remaining motion information can then be used to apply selective search, proposing moving objects.

5.3.2.2 Selective Search on Background Subtracted Images

A slightly more sophisticated approach than using difference images is to first estimate the background of the sequence and then perform selective search on the background subtracted images. The algorithm by Collins et al. has been used for background estimation in this thesis.

The slightly adapted definition from [19] is given in Equation 5.2. The algorithm models the background B as the local temporal average of intensities. Additionally a pixel-wise threshold T_I is maintained that models the local temporal standard deviation. Both are estimated by iterating over the sequence once and updating them every time a new image is presented.

$$\begin{aligned}
\text{stationary iff } T_I(x,y,t) &\leq |I(x,y,t) - I(x,y,t-1)| \wedge |I(x,y,t) - I(x,y,t-2)| \qquad (5.2)\\
T_I(x,y,t+1) &= \begin{cases} bT_I(x,y,t) + (1-b)\,(c\,|I(x,y,t) - B(x,y,t)|) \\ \quad \text{iff } (x,y) \text{ stationary} \\ T_I(x,y,t) \\ \quad \text{else} \end{cases}\\
B(x,y,t+1) &= \begin{cases} aB(x,y,t) + (1-a)\,I(x,y,t) \\ \quad \text{iff } (x,y) \text{ stationary} \\ B(x,y,t) \\ \quad \text{else} \end{cases}
\end{aligned}$$

In each pass, the threshold is compared with the change of a pixel's intensity with respect to the same pixel in the proceeding image and the image before the proceeding image. A pixel is assumed stationary when its intensity change is lower than the threshold and moving otherwise. After locating stationary and moving pixels for the current image, threshold and estimated background are updated based on this information. Both threshold and background remain unchanged for moving pixels.

For stationary pixels however, the threshold is the weighted sum of the old threshold and the difference between the pixels intensity and the background. The background is the weighted sum of the old value and the moving pixel's intensity.

Three hyper-parameters a, b and c can be adapted to tune the algorithm. Parameters a and b define the algorithms sensitivity regarding new information for updating background and threshold, respectively. Parameter c controls the number of standard deviations used by the threshold model.

5.3.2.3 Parameter Optimization and Evaluation

There are two parameters that determine the region proposals generated by selective search. As already described above the *scale* parameter controls the preference for smaller or larger regions whereas the *sigma* parameter controls the smoothing of the Gaussian kernel applied prior to searching for regions. It is important to carefully select these parameters, because the region proposal algorithm should serve as a weak classifier, which means that the selected subset of regions may contain many false positives, but should also contain the regions comprising smoke with high probability. As it will be employed in a cascaded fashion, any smoke region not detected by this algorithm can not be detected at all. Therefore, parameter optimization was performed for all selective search variants that have been considered.

To find the best parameters random grid search in combination with a 3-fold cross-validation was used. Random grid search is a widely used algorithm for parameter optimization and is more effective than an exhaustive grid search. This is due to the fact that not all parameter dimensions have an equal influence on the performance of the algorithm to be optimized.[13]

The objective function that was utilized for optimizing the algorithms was defined as the number of ground truth hits for a given number of sequences. As already described in section 4.2 one or multiple rectangles are associated with each sequence defining the ground truth.

In the literature related to object detection the criterion for a hit is usually defined as an intersection-over-union (IoU) of at least 0.5 (see Equation 5.3).[28] However, as already discussed in section 4.2 the ground truth rectangles at hand are a rather rough estimation for the smoke boundaries. Furthermore, it is less important to capture the whole smoke, so that the criterion for defining whether a given region is a hit or not acknowledges these facts in that it was defined relatively weak.

Selective Search Variant	Hits	Std. Deriv.	Best Parameters	
			Scale	Sigma
Single images	27.62%	5.87%	17	2.5
Difference images	74.29%	1.17%	1	10
Background subtracted images	70.68%	2.76%	9	7.5

Table 5.2: Random grid search results for the selective search variants, the given standard deviation is with respect to the folds of the cross-validation.

The criterion applied here for determining whether a proposed region is a hit or not requires that both the size of the intersection of the ground truth rectangle and the proposed region with respect to the region's size, referred to as intersection over region (IoR), is greater than 0.75 and the size of the intersection of both rectangles with respect to the ground truth's size, referred to as intersection over ground truth (IoG), is greater than 0.2. The formal definition for this criterion is given in Equation 5.6.

$$IoU = \frac{|g \cap r|}{|g \cup r|} \tag{5.3}$$

$$IoR = \frac{|g \cap r|}{|r|} \tag{5.4}$$

$$IoG = \frac{|g \cap r|}{|g|} \tag{5.5}$$

$$\text{Hit} \quad \text{iff} \quad IoR > 0.75 \wedge IoG > 0.2 \tag{5.6}$$

The first part of the criterion requires that the given rectangle lies at least to 75% in the rectangle defined by the ground truth and the second part further restricts regions to at least cover a fifth of the ground truth. This value is so low because it allows small regions that completely lie within the ground truth to be considered a hit. There can be at most one hit for each ground truth rectangle no matter how many of the proposed regions fulfill the criterion.

Discrete linear parameter spaces have been used for both *scale* and *sigma* parameters. For the scale parameter 150 values in the range from 1 to 300 and for sigma 20 values linearly distributed between 0.5 and 10 have been defined, totaling to 3000 possible values. The random grid search was performed on 210 randomly selected sequences from the training sequence set for 300 iterations. The results of the parameter search are given in Table 5.2. They show that selective search on single images is inferior compared to the other variants.

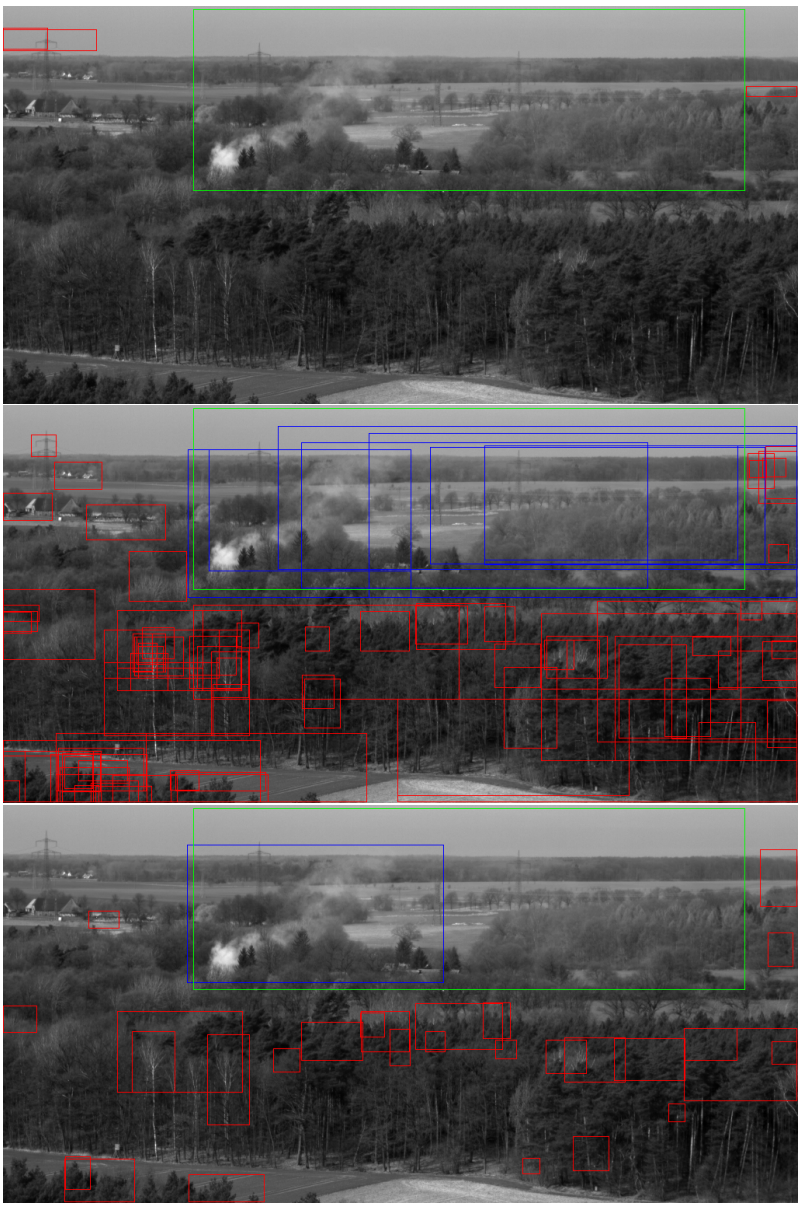

Figure 5.1: Example of region proposals produced by the three selective search variants. Top: single images, middle: difference images, bottom: background subtracted images. The green rectangle defines the ground truth, red and blue rectangles are region proposals whereas blue regions are considered positives and red negatives.

The optimized selective search variants were then further evaluated on the training and validation sequence sets to measure their performance. Besides the relative number of hits, the average number of proposed positive and negative regions were measured, as well as the maximum number of negative regions proposed for any sequence. These additional numbers are of interest, since the number of proposed

Selective Search Variant	Hits	∅ No. pos.	∅ No. neg.	Max. neg.
Single images	23.57%	0.92	46.11	1105
Difference images	68.54%	2.96	63	572
Background subtracted images	74.4%	3.49	72.53	640
Ensemble	83.56%	7.37	181.61	1516

Table 5.3: Results of the selective search variants on all sequences in the *Germany-1835* training and validation sequence sets.

negatives should be as low as possible. Table 5.3 shows the results of the parameter optimized region proposal algorithms. The numbers differ slightly from the results of the random grid search, due to the small number of sequences used for parameter optimization. However, using a greater number of sequences was computationally not feasible.

The results show that the variant that first performs background estimation and then applies selective search on the background subtracted images performs significantly better than the other variants although it produces slightly more negatives compared to the variant working with difference images. Table 5.3 also shows that an ensemble of all three variants would perform even better hitting about 84% of the ground truth. This comes at price, though, since the number of negative proposals also increases dramatically. Probably many of the proposed regions are very similar and could be combined to reduce the overall number of proposals. This was not further investigated, however.

As a result selective search applied on background subtracted images was chosen as region proposal algorithm.

5.3.2.4 Germany-1835 Training and Validation Region Sets

Each positive region proposal for a given sequence was used to extract the image patches at the defined region for all images in the sequence. Table 5.4 lists the number of image patches extracted from the training and validation sequence datasets.

For each negative proposal however, only one image patch was extracted from the whole sequence with the rationale to reduce the number of negative patches and also because an object in a negative region is not changing dramatically over the sequence. Thus the relative number of positives differs in Table 5.4 from the ratio in Table 5.3.

A rectangle with the same aspect ratio as the input size of 200 × 100 was fitted around each region proposal. Additionally, a margin of 10% at the top and left was added to each positive region, to be used to create crops. Effectively, 20 pixels on the left and 10 pixels at the top were added, which would allow the creation of 200 crops. However, in order to balance the datasets only three random crops and their mirrored versions were used.

#Positives	#Negatives	#Total
Training Region Set		
18176 (14.74%)	105095 (85.26%)	123271
Validation Region Set		
4379 (13.52%)	28005 (86.48%)	32384

Table 5.4: Number of data points for training and validation prior to balancing based on the region proposals from selective search on background subtracted images for CNNs.

C3D Region Datasets		
#Positives	#Negatives	#Total
Training		
7043 (4.03%)	167751 (95.97%)	174794
Validation		
1836 (2.83%)	63059 (97.17%)	64895

Table 5.5: Number of data points for training and validation prior to balancing based on the region proposals from selective search on background subtracted images for C3D models.

The region datasets that were used to train the C3D models were created similarly. However, in this case the patches from all images in the sequence for a given region are used to serve as negative example. When a sequence contained more than three images all sub-sequences of length three were used, which increased the number of positive and negative examples.

Since for some sequences there are no positive regions the number of positives did not increase proportional to the number of negatives, requiring an increased number of crops to balance the datasets. Thus 12 random crops and their mirror images were created from each positive example. The same input size as for the region datasets of the CNNs was used and a rectangle with the appropriate aspect ratio was fitted around each region proposal. The detailed numbers of the region datasets used to train the C3D models are shown in Table 5.5.

5.4 Tested Model Architectures

The architecture of a CNN can be defined in terms of number of layers – the model's depths – and number of feature maps per layer. Unfortunately, only rules of thumb

exist for choosing these hyper-parameters. Further hyper-parameters like the stride and filter sizes in convolutional layers, pooling size, *dropout* ratio and type of activation function have to be chosen, creating a huge parameter space. Consequently, models that have proven to be effective for object recognition have been considered in this work. Basically two different models were used as basis.

5.4.1 Architectures for Detection in Still Images

The first model is *CaffeNet* which is a very slight adaption of *AlexNet* which was proposed by Krizhevsky et al. in [46]. The only difference between the two models is that in *CaffeNet* normalization is done after pooling while in *AlexNet* normalization is performed before pooling. The innovation of this model was its deep architecture and the usage of ReLUs and *dropout* and led to new state-of-the-art results for the ImageNet LSVRC-2010 competition.[46]

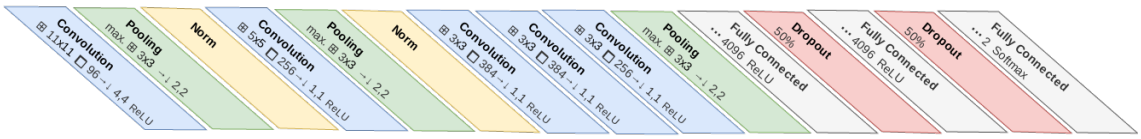

Figure 5.2: The *CaffeNet* model which is an adaption of *AlexNet* (⊞ denotes the size of a layer's kernel, □ gives the number of output feature maps and →↓ lists the stride for the kernel for both dimensions).

The architecture of this model is very straight forward and is shown in Figure 5.2. It consists of five convolutional layers all using ReLUs, max-pooling and normalization layers. Three fully-connected layers are used to classify the feature maps of the last convolutional layers. The fully-connected layers also employ ReLUs. Furthermore *dropout* is applied to the first full-connected layer.

Two variants of *CaffeNet* have been considered. The first is the original *CaffeNet* and the second is $CaffeNet_{red}$ where the number of feature maps at each layer is quartered, so that the complexity of the model is reduced. The idea behind this was to improve generalization, since *CaffeNet* was originally used to predict 1000 classes and is thus probably over-complex.

The second model is a more recent development by Szegedy et al. called *GoogLeNet*.[71] Its architecture is significantly different to "classical" CNNs like *CaffeNet*, although it uses the same basic building blocks. As already described in section 3.2 *GoogLeNet* consists of inception modules that implement a sparse connection scheme using fully-connected layers. The motivation for this architecture is based on the findings of Arora et al., who showed that under certain conditions optimal network architectures can be constructed layer by layer when following a sparse connection scheme.

The original *GoogLeNet* is made of three regular convolutional layers close to the input followed by nine inception modules each consisting of six convolutional layers. The total number of layers is therefore huge, even when counting layers with learnable parameters only.

Training the original *GoogLeNet* on the available hardware was not possible due to lack of GPU memory. Therefore smaller versions of *GoogLeNet* regarding the number of inception modules and feature maps have been used instead. A variant with only a single inception module but with a higher number of feature maps shown in Figure 5.3 and a variant with two inception modules but reduced number of feature maps shown in Figure 5.4 have been employed. They will be referred to as $GoogLeNet_1$ and $GoogLeNet_2$ respectively.

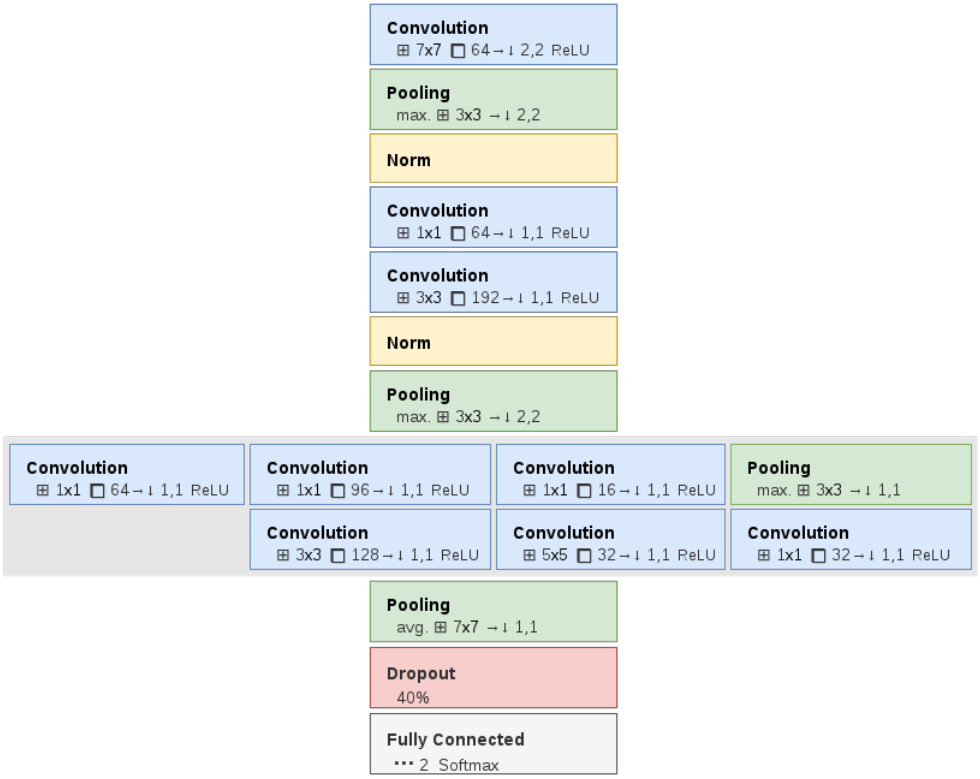

Figure 5.3: $GoogLeNet_1$ with a single inception module (⊞ denotes the size of a layer's kernel, ☐ gives the number of output feature maps and →↓ lists the stride for the kernel for both dimensions).

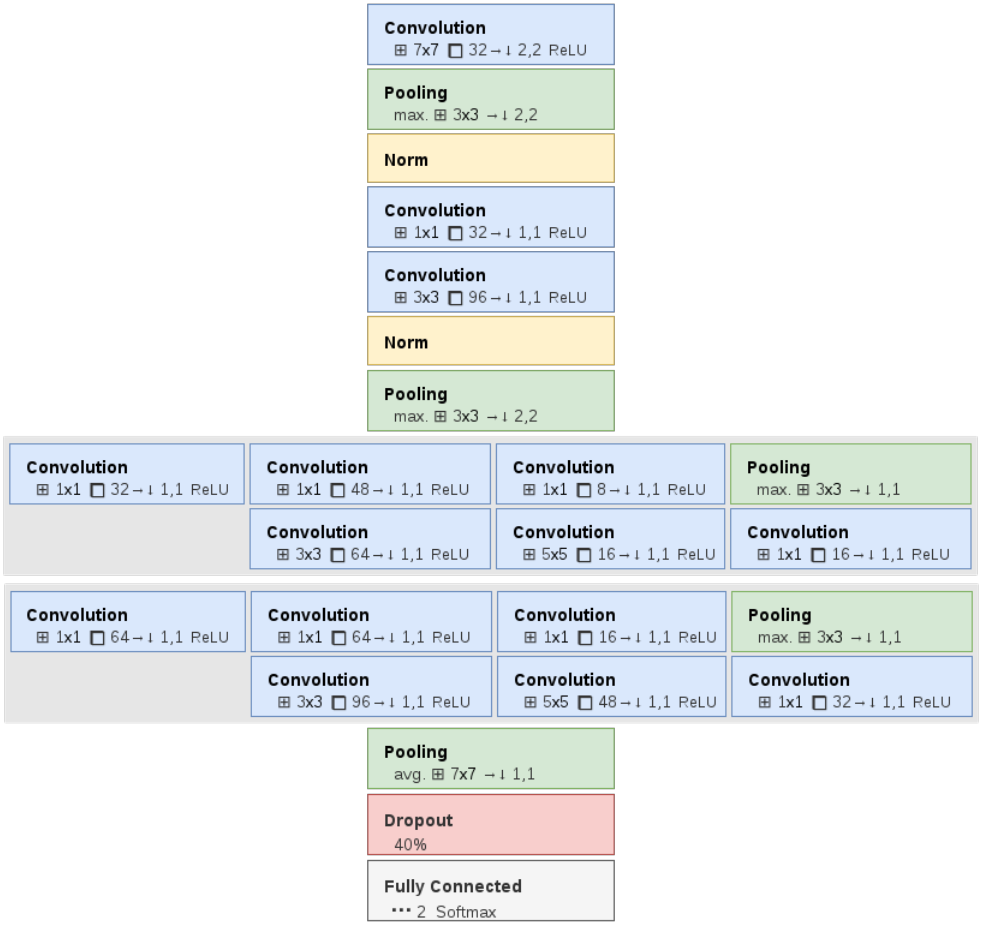

Figure 5.4: *GoogLeNet*$_2$with two inception modules but less feature maps (⊞ denotes the size of a layer's kernel, ☐ gives the number of output feature maps and →↓ lists the stride for the kernel for both dimensions).

5.4.2 Architectures using Spatio-temporal Features

Since the specific motion of the smoke plume can be considered an important feature, models that can also learn features in the temporal dimension have been investigated in addition to the models working on still images. As the natural extension of CNNs C3D models were used in this regard. C3D models use three-dimensional rather than two-dimensional filters to extract features from the given data, which allows learning on a sequence of images.

C3D models have been applied to several problems dealing with sequential image data, yet no specific architectures that differ from those used in the field of still image recognition have evolved (see subsection 3.2.3). Hence, the same models that were used for still image recognition have been used as basis for two different C3D models. An architectural aspect that could be considered is the point of fusing the

temporal information as described in [44], i.e. extracting temporal features. The authors find that it is beneficial to include the temporal information slowly through the network rather than consuming it completely at the beginning or at the end. Though, since the sequences in the used dataset have only a length of three, at most two convolutions with the minimum temporal kernel dimension of two could be used. However, Karpathy et al. also state that a minimum size of three should be used, so that the tested models all use early fusion.

The two C3D models that were trained are based on the $CaffeNet_{red}$ model with reduced complexity and the $GoogLeNet_{2}$ model with two inception modules. The filter sizes of the first convolutional layer in both models were changed to three for the temporal dimension so that they perform early fusion of the temporal information. Subsequent layers were not changed. These models are labeled *C3D* $CaffeNet_{red}$ and *C3D* $GoogLeNet_{2}$.

6 Results

This chapter describes the results for the trained models first defining which measures were used. Since the sliding window approach resulted in too bad performance, no detailed results are given. Instead it is focused on the evaluation of the models trained on selective search region proposals.

6.1 Evaluation Measures

When evaluating models for binary classification on a given dataset of positives and negatives usually four different types of data are defined: True positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), whereas true and false refers to whether the positives or negatives were correctly classified by the predicting model. A perfect classifier would yield no FN and FP, since all positives and negatives would be predicted correctly.

Based on the numbers of TP, TN, FP and FN a variety of measures exists to benchmark a given model. For the wildfire detection task not all of these measures are meaningful or suitable, since there is a great imbalance between the number of negatives and positives. Every measure that relates numbers of the different classes, can therefore be misleading.

As in the case of dataset generation, a criterion has to be defined to decide whether a region is considered a positive or a negative. Here, the same criterion as defined in subsection 5.3.2 Equation 5.6 is used. It considers a region as positive when its intersection with the ground truth in relation to the region's size is greater than 75% and the intersection regarding the ground truth's size is greater than 20%.

Two measures are most important for smoke detection. First, all occurrences of smoke should be detected or equivalently no smoke should be missed. This is measured by the relative number of TP with respect to the overall number of positives and is called the true positive rate (TPR, also recall, sensitivity or hit rate). Its definition is given in Equation 6.1.

Because every time the model detects smoke, an alarm is sent to the control center, the number of false alarms or FP should be as low as possible. The false positive rate (FPR, also fall-out) can be used to measure this. It is the relative amount of FP regarding the overall number of negatives (see Equation 6.2).

Another, however not so meaningful, measure is accuracy (ACC). It is an overall measure for the relative number of correct classifications of both positives and negatives and is defined in Equation 6.3. The issue with this measure for the smoke detection problem is that the number of negatives is much larger than the number of positives. However, it can still be used to compare the overall performance of the different models.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{6.1}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{6.2}$$

$$ACC = \frac{TP + TN}{P + N} \tag{6.3}$$

As the definition of the smoke detection problem actually refers to sequences, but all models are applied on regions, the performance can be also measured with respect to regions or sequences. The measures that are used to evaluate these two differ, however.

Region based evaluation is straight forward. A given region is either positive or negative using the criterion already described. Consequently, TPR, FPR and ACC can be calculated easily.

When it comes to sequences evaluation is more difficult. Usually many regions are predicted for a given sequence. Some of these regions are positives if the sequence itself is a positive and some regions are negatives. In the simplest case all positive and negative regions would be correctly classified or none of the positive regions would be detected, so that the sequence would be considered a TP in the first and a FN in the latter case. With the presence of FP and FN regions things become more complicated. If for a positive sequence at least one TP region exists, it should be counted as a TP sequence, because some smoke in the sequence was correctly detected.

False alarms play an important role for smoke detection. Therefore it is desired to measure how many sequences have been falsely considered containing smoke. Thus, if there is a FP region, a sequence should be counted as FP, even if also a TP region exists.

Since a sequence could be a TP and a FP, the sum of FP and TP is not equivalent to the overall number of positives anymore. As a result the definition of the FPR is adapted for the evaluation of sequences as defined in Equation 6.4. Instead of the relative number of FP regarding the overall number of positives, it is now defined

as the relative number of FP with respect to the whole number of sequences.

$$FPR_{Seq} = \frac{FP}{P + N} \tag{6.4}$$

For any results a subscript indicates if the given measure applies to region or sequence based evaluation. For example TPR_{Reg} indicates the TPR for region based evaluation while TPR_{Seq} refers to the TPR regarding the sequences.

The receiver operating characteristic (ROC) is a diagram that plots FPR against TPR for varying thresholds. It is a widely used plot for the evaluation of binary classifiers. ROC curves will therefore be used for evaluation too when appropriate.

The trained models will be compared to the *fshell* algorithm. However, it is important to keep in mind that the data that was used is biased by the *fshell* algorithm, because data is received for wildfires that were detected by the algorithm only. In order to decrease this bias, sequences that were captured chronologically before and after a given sequence have been included. This does not apply to the *Löcknitz-Negative* dataset of course, since this dataset does not contain any positives.

6.2 Sliding Window

The first approach to the detection problem was to scan each image in a sequence using different window sizes. In the training phase random samples where therefore used as negative examples.

Apparently, this did not work well, because the trained networks were very sensitive and produced a lot of FP. One reason is that the number of regions that is examined is very large. Scanning an image of the size 1024×512 at a window size of 100×50 with a stride of 50% of the window's size produces already about 400 regions, so that the FPR_{Reg} has to be very low. Considering the large number of regions when scanning an image, the number of randomly sampled negatives that was used for training has been probably too small. However, a training dataset with 1,500 sequences and 400 regions per sequence would contain over 600,000 image patches, which is not impossible to handle but makes training very time consuming.

Due to the poor performance, the models trained on randomly sampled image patches were not further evaluated, instead I focused on the more promising approach using selective search.

6.3 Selective Search

By using selective search on background subtracted images as a weak classifier that pre-selects regions where motion is present, the models only had to deal with a

sub-class of negatives instead of any possible region.

Evaluation was done on the *Germany-1835* and *Löcknitz-Negative* datasets. In addition to the single model performance an ensemble of all models was evaluated. The ensemble uses the average of the predictions of all the models.

6.3.1 Germany-1835

The *Germany-1835* dataset contains 1,835 sequences each showing at least one wildfire. The subset of this dataset that was used for training contains 1,525 sequences and the set for validation 310 sequences. Due to the performance of the utilized selective search variant, the maximum number of detectable sequences for the models is reduced to 72% and 76% for training and validation respectively.

Figure 6.1 shows the ROC curves for the TPR and FPR with respect to the classification of regions for the training and validation sets. In each plot the upper bound defined by selective search is marked by a gray dashed line. Since the curves for the different models are very close, the FPR and TPR have been plotted from 0 to 0.2 and from 0.85 to 0.92, respectively, otherwise distinguishing the curves would be impossible.

Anyhow, in the training case, the curves for the C3D models and the ensemble still almost completely overlie each other, suggesting that the ensemble's performance is limited by the performance of the C3D models, that are the best performing models. The plots also show that $GoogLeNet_1$ and $GoogLeNet_2$ perform worse than the other models regarding the TPR except for the validation set where the TPR of the *C3D* $CaffeNet_{red}$ stagnates for a FPR greater 0.08.

All in all this results are already very promising. For the validation set, which was not used for training, the C3D models and the *CaffeNet*s are able to reach a TPR of over 90% while keeping the FPR below 3%.

Looking at the ROC curves for the evaluation on the sequences in Figure 6.2, the performance gap between the C3D models and the CNN models is even more apparent. Here, the ROC is not zoomed. However, the diagrams also show that reaching a TPR of 70% is not feasible, because the FPR would be too high. A peculiarity is that the deep $GoogLeNet_2$ is the worst performer while its C3D counter part yields the best results. It seems the deep architecture is able to extract better features when provided with the additional temporal information.

From the ROC curves one can see which TPR comes at what price in terms of FPR. However, it is not possible to determine which threshold underlies a certain data point in this plot. To do so it is necessary to plot the threshold against the FPR and TPR which was done for the model ensemble and both validation and training datasets. It is shown in Figure 6.3. The TPR is only slowly reduced when the threshold is increased.

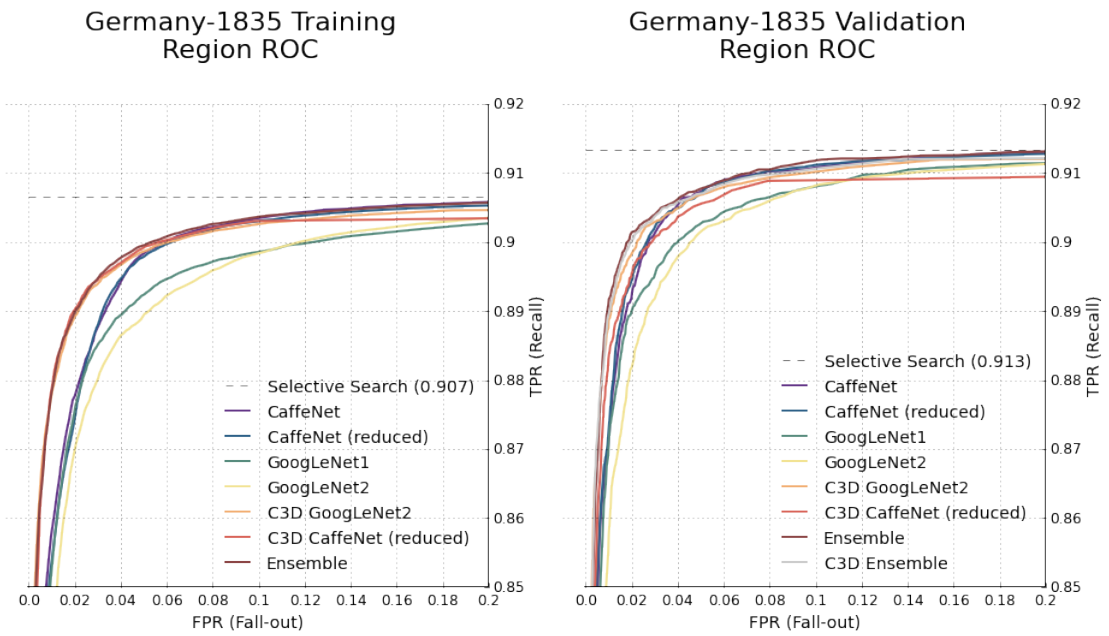

Figure 6.1: Zoomed upper left parts of the ROC curves for the *Germany-1835* training and validation datasets regarding the classification of regions.

The plot indicates that a threshold of 0.8 seems to be a good choice retaining a high TPR while keeping the FPR low. With the threshold of 0.8 the performance of the models was measured regarding the ACC, TPR and FPR for both regions and sequences. The results are given in Table 6.1.

Due to the *fshell*'s functionality it does not classify negative regions, but gradually refines a selection of regions based on predefined filters. As a result no evaluation can be done on region basis.

The result show that the ACC, TPR and FPR are for the regions seem to be good. However, the numbers of the sequence based evaluation show that the FPR is an issue and that the TPR of *fshell* is not reached.

Nonetheless, the results show clearly what was already indicated by the ROC curves. The models that can make use of temporal features have a TPR comparable to the best non-temporal models while the FPR is much lower. The ensemble shows the best FPR, but also slightly reduced TPR.

Figure 6.4 shows the FN regions with the largest error. The images emphasize the difficulty of the smoke detection problem. Many FN examples show only a small fraction of the actual smoke plume which is probably due to few motion. These image details are indeed positives, but are very hard to classify without more context. Adding extra context, i.e. a larger rectangle, to a given region proposal might help in this case.

There are also a few examples where windmills were detected and the region also

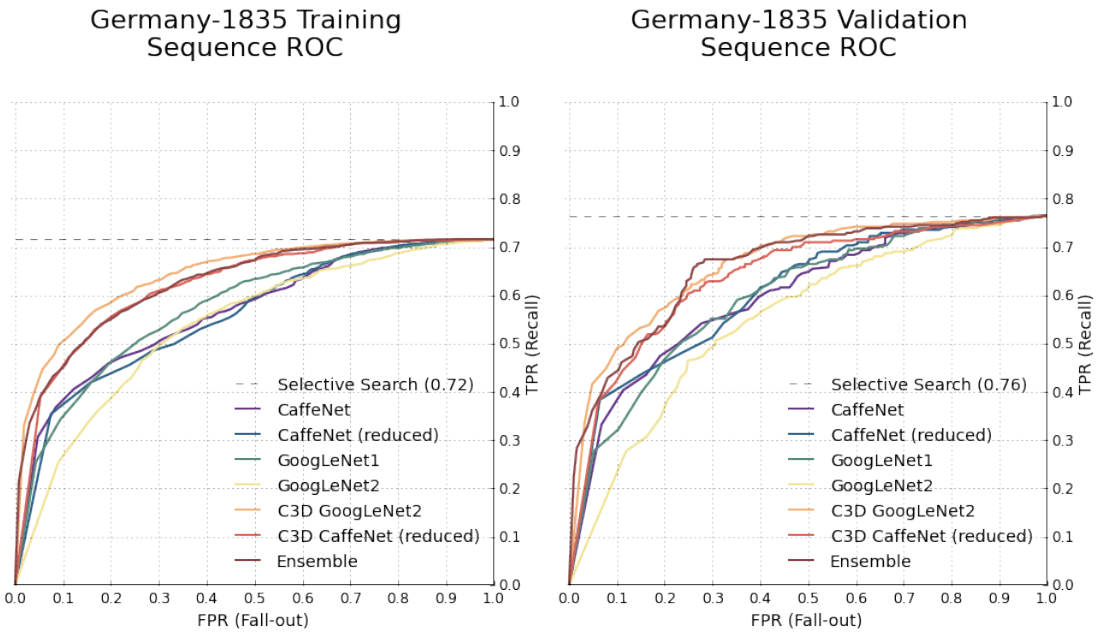

Figure 6.2: ROC curves for the performance regarding sequence classification for the *Germany-1835* training and validation datasets.

happens to lie in the ground truth. This can be seen in Figure 6.5 where context is given for the first and third FN examples.

Looking at the FP with the largest errors, similar problems exist. Most of the FP are indeed misclassifications like clouds or misty weather, but there are also examples where smoke was correctly detected, but ground truth was missing.

6.3.2 Löcknitz-Negative

The *Löcknitz-Negative* dataset contains sequences that were captured from a single tower. It is split in three parts, where each part corresponds to a direction the camera is pointed at, called the bearing. There are 48 sequences for the bearings 224° and 37° and 47 sequences for the 312° bearing.

All sequences do not contain smoke and were taken on days where the *fshell* algorithm produced many false alarms. Playing the sequences reveals the reason for this large number of false alarms quite clearly. There is a lot of motion, because of strong wind. Furthermore weather changes and cloud movement induce fast lighting changes and shadows. In the 224° direction there are also many windmills that are constantly moving and whose bases have some similarity with smoke regarding the white color.

Table 6.2 lists the results for the different directions. Although, the FPR for the regions is not extremely high, the FPR for the sequences is almost 100% for the first

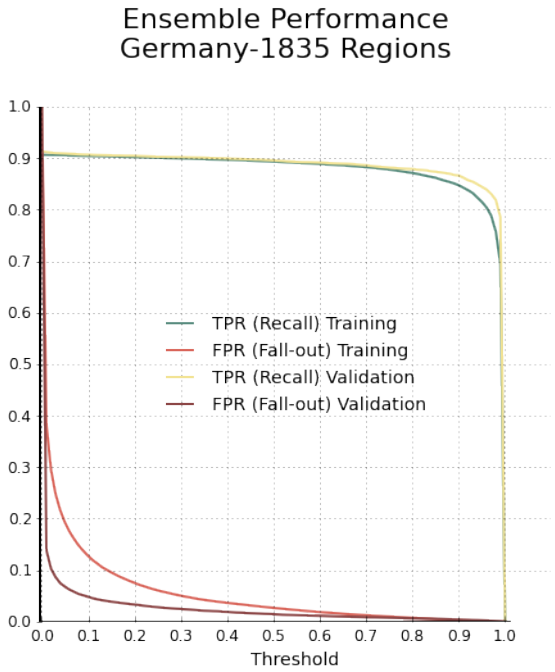

Figure 6.3: Zoomed upper left parts of the ROC curves for the *Germany-1835* training and validation datasets regarding the classification of regions.

direction. Due to the motion in the sequences, the number of regions proposed by selective search increases. In the case of the first direction the average number of regions is 139, so that statistically there is at least one FP region in each sequence leading to the very high number of FP sequences.

Again, the *C3D GoogLeNet*$_2$ shows the best performance on average. The *fshell* algorithm shows a much better FPR for the first two directions, only in the case of the last direction it is topped by the ensemble, which also has a much lower FPR than any single model. This suggests that combining multiple models is crucial for a low FPR.

For each bearing a heat map was created that indicates regions where many false alarms were raised

For the 224° bearing the most FP regions are located in the upper right corner as shown in Figure 6.8 at the top. This is probably due to the white windmill bases towering from below the horizon above it.

The heat map for the direction of 312° in the middle of Figure 6.8 shows that the FP are more evenly distributed over the whole region above the horizon. The models seem to have responded to the moving clouds here.

In the case of the last direction of 37°, clouds seem to be causing many FP, too. Detecting the horizon and excluding the region above it might therefore help to significantly reduce the number of FP, which is also done by the *fshell* algorithm.

Model	ACC_{Reg}	TPR_{Reg}	FPR_{Reg}	TPR_{Seq}	FPR_{Seq}
Training					
fshell	NA	NA	NA	76.00%	8.00%
CaffeNet	97.57%	87.77%	2.01%	58.69%	48.72%
$\text{CaffeNet}_{\text{red}}$	97.06%	88.30%	2.54%	61.97%	54.62%
GoogLeNet_1	97.71%	87.30%	1.85%	59.02%	40.33%
GoogLeNet_2	98.14%	85.61%	1.40%	52.33%	34.03%
C3D GoogLeNet_2	98.53%	87.79%	1.00%	61.84%	25.77%
C3D $\text{CaffeNet}_{\text{red}}$	98.50%	87.94%	1.03%	60.85%	29.64%
Ensemble	98.79%	87.13%	0.73%	57.05%	23.41%
Validation					
fshell	NA	NA	NA	75.00%	7.00%
CaffeNet	98.18%	88.62%	1.49%	65.48%	50.97%
$\text{CaffeNet}_{\text{red}}$	97.99%	89.19%	1.69%	68.71%	51.94%
GoogLeNet_1	98.29%	87.96%	1.38%	63.55%	42.90%
GoogLeNet_2	98.53%	86.44%	1.12%	55.16%	38.39%
C3D GoogLeNet_2	98.81%	88.44%	0.84%	62.90%	27.42%
C3D $\text{CaffeNet}_{\text{red}}$	98.96%	88.14%	0.97%	60.97%	23.23%
Ensemble	99.00%	87.90%	0.64%	59.03%	19.68%

Table 6.1: Results for the *Germany-1835* training and validation datasets using a threshold of 0.8 for all models.

Furthermore, regions where treetops are lightened under certain conditions also seem to produce many FP in the last case.

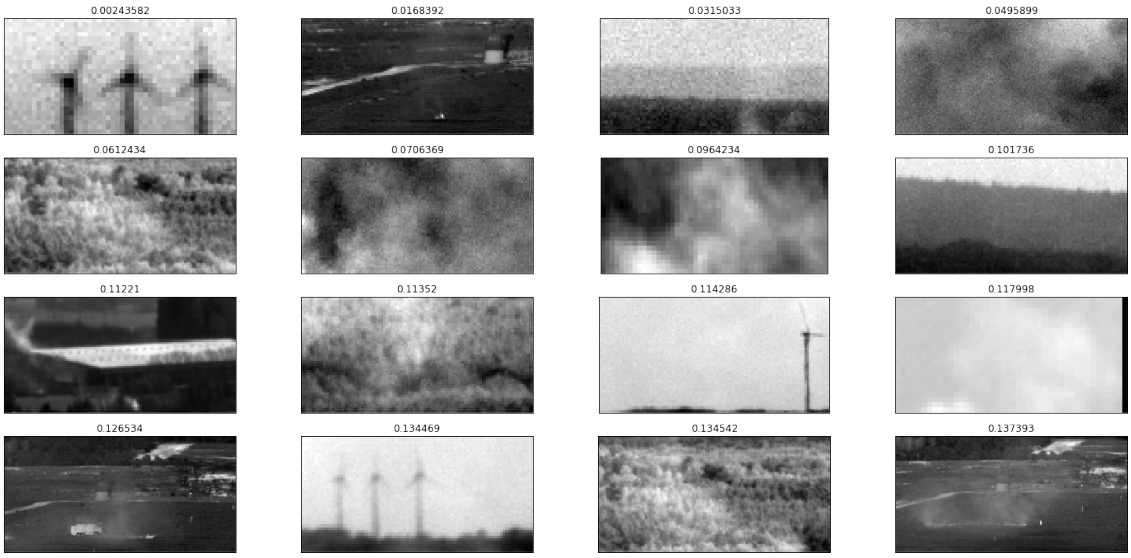

Figure 6.4: False negative regions from the *Germany-1835* validation dataset with the predicted smoke probability above each image.

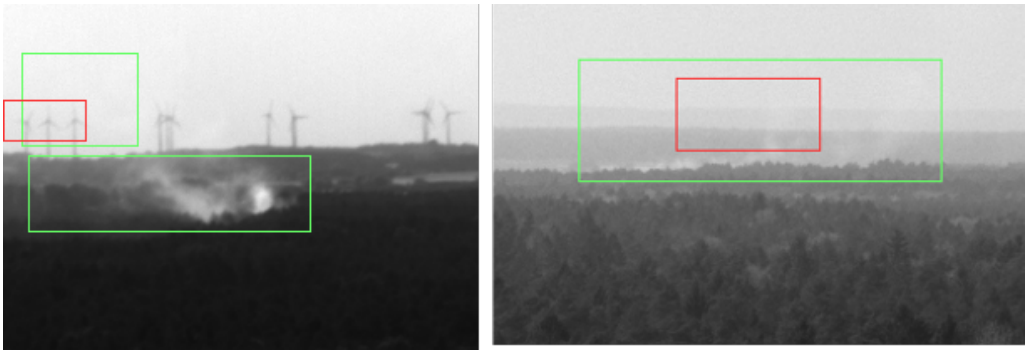

Figure 6.5: False negative regions from the *Germany-1835* validation dataset in a larger context. On the left a region enclosing windmills happens to lie in the ground truth. On the right the proposed region seems to be to small to decide whether smoke is present or not.

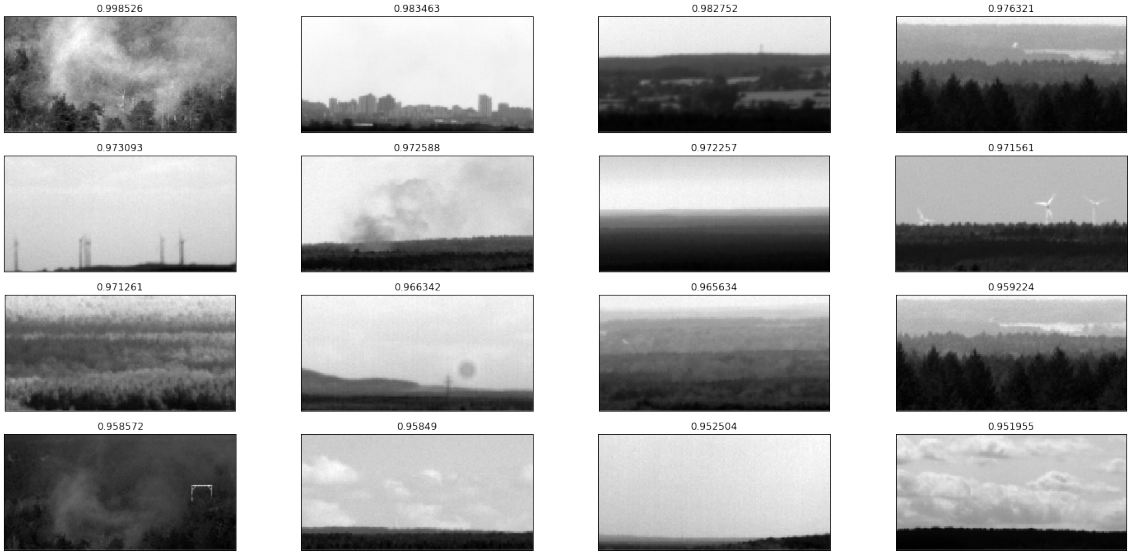

Figure 6.6: False positive regions from the *Germany-1835* validation dataset with the predicted smoke probability above each image.

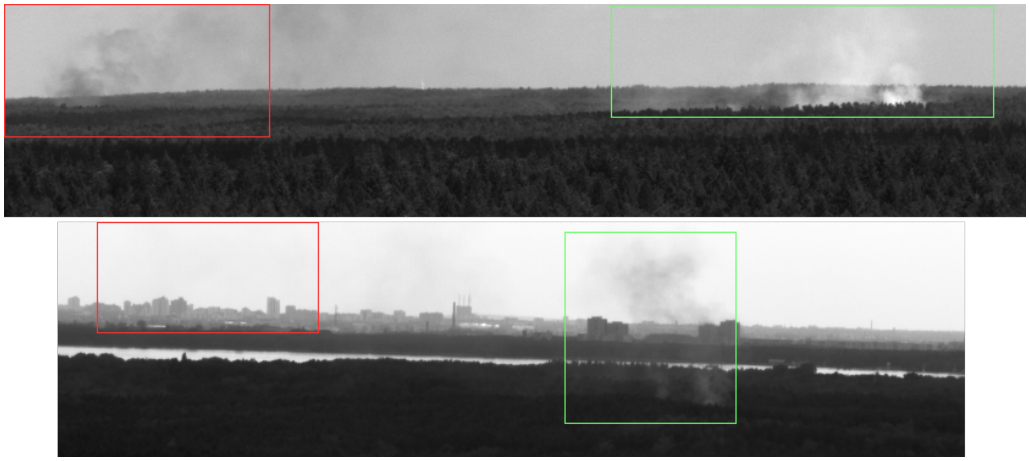

Figure 6.7: False positive regions from the *Germany-1835* validation dataset in a larger context. Top: smoke was detected that was not marked. Bottom: actual misclassification.

	224°		312°		37°	
Model	FPR_{Reg}	FPR_{Seq}	FPR_{Reg}	FPR_{Seq}	FPR_{Reg}	FPR_{Seq}
fshell	NA	41%	NA	11%	NA	52%
CaffeNet	9.33%	97.92%	6.04%	87.23%	1.48%	72.92%
$\text{CaffeNet}_{\text{red}}$	7.57%	97.92%	6.57%	89.36%	1.74%	64.58%
GoogLeNet_1	6.94%	97.92%	6.41%	91.49%	2.39%	81.25%
GoogLeNet_2	6.48%	97.92%	6.30%	93.62%	2.20%	85.42%
C3D GoogLeNet_2	4.69%	87.50%	3.99%	78.72%	1.96%	72.92%
C3D $\text{CaffeNet}_{\text{red}}$	7.80%	97.92%	4.63%	89.36%	2.29%	79.17%
Ensemble	3.88%	87.50%	3.67%	78.72%	0.96%	45.83%

Table 6.2: *Löcknitz-Negative* Results

Figure 6.8: Heat map for the location of misclassified regions in the direction of 224° (top), 312° (middle) and 37°(bottom). Pixels that are part of a larger number of FP regions are marked with higher red intensities.

7 Conclusion

In this thesis I showed that the usage of CNNs for wildfire smoke detection is a very promising approach to increase the effectiveness of camera based automated detection systems. The trained models are able to detect 88% of regions containing smoke at an FPR of about 1%. Considering sequences rather than regions, 60% of all sequences containing smoke are recognized while about 23% false alarms are created.

Two major insights have been gained. First, a region proposal algorithm should be employed to reduce the number of regions that have to be classified. It was shown that a variant of the selective search algorithm that is applied to background subtracted images can be used to propose regions where motion is present, hitting a smoke region in 74% when a sequence contains smoke, while proposing 73 negative regions per sequence on average. More sophisticated region proposal algorithms could be investigated, since this is the bottleneck for the whole pipeline. However, the focus of this thesis was on investigating the general applicability of CNNs rather than finding an optimal region proposal algorithm.

Second, neural network models that are able to leverage temporal information yield a significantly better performance than CNNs that use spatial information only. In this regard C3D networks have been trained and evaluated. They showed a FPR that was 10 to 20% lower compared to regular CNNs.

The performance of the trained models is indeed lower compared to the *fshell* algorithm. However, the CNNs completely rely on image data only whereas *fshell* can make use of additional features like distance information derived from the tower height and a noise model of the camera.

Future work could consider a variety of mechanisms to further improve the performance of CNNs for wildfire detection. Adding information about the distance and size to the inputs of a CNN would allow to learn a relation between these two features, leading to a lower FPR, since the model could learn to classify small regions in the foreground with a bias towards the class of negatives similar to the functionality of *fshell*.

Furthermore, a more sophisticated weak classifier than the used selective search variant could be employed, reducing the number of proposed regions while increasing the hit rate for regions containing smoke.

Generally, the use of an ensemble is also very promising. For example training a combination of CNN and C3D models in a boosting fashion might lead to a greatly

reduced FPR.

Bibliography

[1] Computer vision based fire detection software. http://signal.ee.bilkent.edu.tr/VisiFire/. Accessed: 2015-08-16.

[2] Wildfire observers and smoke recognition homepage. http://wildfire.fesb.hr/. Accessed: 2015-08-16.

[3] Ostrogradski formula. http://www.encyclopediaofmath.org/index.php?title=Ostrogradski_formula&oldid=31341. Accessed: 2015-08-15.

[4] Torch - a scientific computing framework for luajit. http://torch.ch/. Accessed: 2015-08-28.

[5] Jarmo Alander, Timo Honkela, and Matti Jakobsson. Turing machines are recurrent neural networks. In *Publications of the Finnish Artificial Intelligence Society*, pages 13 – 24, 1996.

[6] Alfonso Alba, Ruth M. Aguilar-Ponce, Javier Flavio Vigueras-Gómez, and Edgar Arce-Santana. Phase correlation based image alignment with subpixel accuracy. In Ildar Batyrshin and Miguel González Mendoza, editors, *Advances in Artificial Intelligence*, volume 7629 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2013.

[7] Ahmad A. A. Alkhatib. A review on forest fire detection techniques. *International Journal of Distributed Sensor Networks*, 2014, 2013.

[8] Melih Altun and Mehmet Celenk. Smoke detection in video surveillance using optical flow and green's theorem. In *WorldComp 2013 Proceedings*, 2013.

[9] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343, 2013. URL http://arxiv.org/abs/1310.6343.

[10] Thomas Behnke, Hartwig Dr. Hetzheim, Herbert Prof. Jahn, Jörg Dr. Knollenberg, and Ekkehard Dr. Kührt. Verfahren und vorrichtung zur automatischen waldbranderkennung, 2006.

[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *ArXiv e-prints*, June 2012.

[12] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A cpu and gpu math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[13] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)*, 13, 2012.

[14] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995.

[15] R. Bogush and N. Brovko. An efficient smoke detection algorithm for video surveillance systems based on optical flow. In *Pattern Recognition and Information Processing*, 2011.

[16] Landesbetrieb Forst Brandenburg. Waldreiches land - bestockungsverhältnisse. `http://forst.brandenburg.de/sixcms/detail.php/631357`. Accessed: 2015-08-10.

[17] Ramón y Cajal. The structure and connexions of neurons. *Nobel Lecture*, 1906.

[18] Yu Chunyu, Zhang Yongming, Fang Jun, and Wang Jinjun. Video smoke recognition based on optical flow. In *Advanced Computer Control (ICACC)*, volume 2, 2010.

[19] Robert Collins, Alan Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, and Osamu Hasegawa. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Pittsburgh, PA, May 2000.

[20] Alejandro Dominguez. A history of the convolution operation. *IEEE Pulse*, January/February 2015. URL `http://pulse.embs.org/january-2015/history-convolution-operation/`.

[21] Howard Eichenbaum. *The Cognitive Neuroscience of Memory: An Introduction*. Oxford University Press, 2011.

[22] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 2004.

[23] David J. Fleet and Yair Weiss. Optical flow estimation. In *Mathematical models for Computer Vision: The Handbook*, pages 239–257. Springer, 2005.

[24] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[25] Bundesanstalt für Ernährung und Landwirtschaft (BMEL). Der wald in deutschland – ausgewählte ergebnisse der dritten bundeswaldinventur, October 2014.

[26] Bundesanstalt für Ernährung und Landwirtschaft (BMEL). Waldbrandstatistik der bundesrepublik deutschland für das jahr 2014, 2015.

[27] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL `http://arxiv.org/abs/1504.08083`.

[28] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL `http://arxiv.org/abs/1311.2524`.

[29] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[30] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *ArXiv e-prints*, February 2013.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014. URL `http://arxiv.org/abs/1406.4729`.

[32] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, July 2006.

[33] Toshiteru Homma, Les E. Atlas, and Robert Marks II. An artificial neural network for spatio-temporal bipolar patters: Application to phoneme classification. *Advances in Neural Information Processing Systems*, 1:31–40, 1988.

[34] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106 – 154, January 1962.

[35] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, 28(2):229–289, March 1965.

[36] D. H. Hubel and T. N. Wiesel. Ferrier lecture: Functional architecture of macaque monkey visual cortex. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 198(1130):pp. 1–59, 1977.

[37] IQ Wireless GmbH. Firewatch - an early warning system for forest fires, successfully in the global use. `http://www.fire-watch.de`. Accessed: 2015-08-25.

[38] Mark Z. Jacobson. Effects of biomass burning on climate, accounting for heat and moisture fluxes, black and brown carbon, and cloud absorption effects. *Journal of Geophysical Research: Atmospheres*, 119:8980–9002, 2014.

[39] Toni Jakovčević, Ljiljana Šerić, Darko Stipaničev, and Damir Krstinić. Wildfire smoke-detection algorithms evaluation. In *VI International Conference on Forest Fire Research*, 2010.

[40] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153. IEEE, 2009. URL `http://dblp.uni-trier.de/db/conf/iccv/iccv2009.html#JarrettKRL09`.

[41] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *Proceedings of the 27th International Conference on Machine Learning*, 35(1):221 – 231, January 2010.

[42] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[43] Eric R. Kandel and Larry R. Squire. Neuroscience - breaking down scientific barriers to the study of brain and mind. *Annals of the New York Academy of Sciences, 2001, Vol.935(1), pp.118-135*, 935(1):118.

[44] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[45] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012.

[47] R. Donida Labati, A. Genovese, V. Piuri, and F. Scotti. Wildfire smoke detection using computational intelligence techniques enhanced with synthetic smoke plume generation. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 43(4):1003 – 1012, July 2013. 2168-2216.

[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2324, November 1998.

[49] Wen-Hui Li, Bo Fu, Lin-Chang Xiao, Ying Wang, and W.Pei-Xun Liu. A video smoke detection algorithm based on wavelet energy and optical flow eigenvalues. *Journal of Software*, 8(1):63–70, January 2013.

[50] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. URL `http://arxiv.org/abs/1312.4400`.

[51] Alan Longstaff. *Neuroscience*. Garland Science, New York, 2011. ISBN 9780203808290 0203808290.

[52] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[53] Elman Mansimov, Nitish Srivastava, and Ruslan Salakhutdinov. Initialization strategies of spatio-temporal convolutional neural networks. *ArXiv e-prints*, 2015.

[54] Stuart Matthews, Andrew Sullivan, Jim Gould, Richard Hurley, Peter Ellis, and John Larmour. Evaluation of three fire detection systems. Technical report, Bushfire Cooperative Research Centre, 2010.

[55] Warren S. McCulloch and Walter H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115 – 133, 1943.

[56] Charles Mercier. Le massif landais, 1974.

[57] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The M.I.T. Press, 1969.

[58] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010. URL `http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10`.

[59] Christopher Olah. Neural networks, manifolds, and topology. `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`, 2014. Accessed: 2015-09-18.

[60] Yair Poleg, Ariel Ephrat, Shmuel Peleg, and Chetan Arora. Compact cnn for indexing egocentric videos. *ArXiv e-prints*, abs/1504.07469, April 2015.

[61] Trygve Randen and John Hå Husøy. Filtering for texture classification: A comparative study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(4):291–310, April 1999. ISSN 0162-8828. doi: 10.1109/34.761261. URL `http://dx.doi.org/10.1109/34.761261`.

[62] Suchet Rinsurongkawong, Mongkol Ekpanyapong, and Matthew N. Dailey. Fire detection for early fire alarm based on optical flow video processing. In *9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2012.

[63] R Rojas. *Neural Networks – A Systematic Introduction*. Springer-Verlag, 1996.

[64] Frank Rosenblatt. The perceptron: A perceiving and recognizing automaton (project para). Technical report, Cornell Aeronautical Laboratory, 1957.

[65] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In *Symposium on Parallel and Distributed Processing*, 1986.

[66] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing*. The MIT Press, 1987.

[67] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015. doi: 10.1007/s11263-015-0816-y.

[68] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In Konstantinos Diamantaras, Wlodek Duch, and LazarosS. Iliadis, editors, *Artificial Neural Networks - ICANN 2010*, volume 6354 of *Lecture Notes in Computer Science*, pages 92–101. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15824-7. doi: 10.1007/978-3-642-15825-4_10. URL `http://dx.doi.org/10.1007/978-3-642-15825-4_10`.

[69] Mubarak Shah. *Fundamentals of Computer Vision*. University of Central Florida, 1997.

[70] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014. URL `http://dl.acm.org/citation.cfm?id=2670313`.

[71] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

[72] F.H.C. Tivive and A. Bouzerdoum. Texture classification using convolutional neural networks. In *TENCON 2006. 2006 IEEE Region 10 Conference*, 2006.

[73] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. *ArXiv e-prints*, abs/1412.0767, December 2014.

[74] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[75] P. Werbos. *Beyond Regression – New Tools for Prediction and Analysis In the Behavioral Science*. PhD thesis, Harvard, 1974.

[76] B. Widrow. An adaptive "adaline" neuron using chemical "memistors". Technical report, Stanford University, Stanford Electronics Laboratories, 1960.

List of Figures