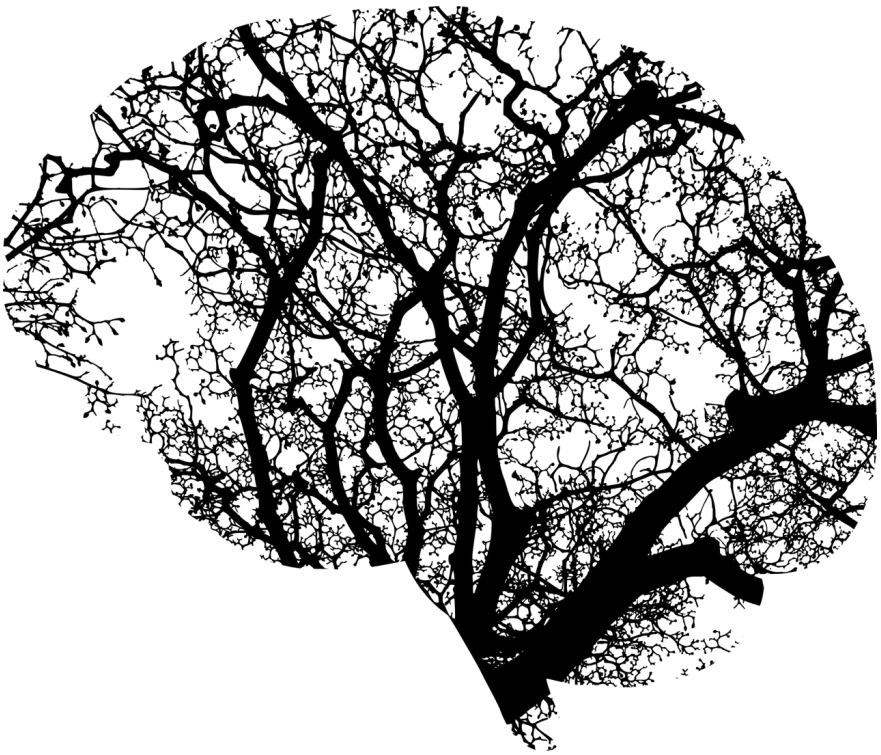
CHALMERS
UNIVERSITY OF TECHNOLOGY



A multimodal deep learning approach for real-time fire detection in aerial imagery

Daniel Posch, Jesper Rask

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

A multimodal deep learning approach for real-time fire detection in aerial imagery

Daniel Posch, Jesper Rask

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

A multimodal deep learning approach for real-time fire detection in aerial imagery.

Daniel Posch, Jesper Rask

Supervisor: Amir Shahroudy, Signal processing and Biomedical engineering
Andreas Björnberg, Carmenta
Examiner: Fredrik Kahl, Signal processing and Biomedical engineering

Master's Thesis 2019
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice

Gothenburg, Sweden 2019

A multimodal deep learning approach for real-time fire detection in aerial imagery
Daniel Posch, Jesper Rask
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering Chalmers University of Technology

Abstract

An increasing wildfire risk is a reality for a big part of the world. Warmer temperatures and drier conditions are the major contributors. The key to control a fire, is to quickly locate the affected area, before it reaches an uncontrollable state. As a consequence, researchers have shown an increased interest in a solution to a reliable early fire detection. Fire departments need an effective solution to locate and map the possible fires in an area. By introducing aerial fire detecting units, the fires could quickly be located and suppressed.

Convolutional neural networks are the current state of the art technique regarding image classification. This thesis explores the possibilities of extracting fire features by trained models, with the purpose of improving the accuracy and detection speed of the traditional approaches. By evaluating known CNN architectures such as VGG16, FireNet, DenseNet and InceptionV3 using transfer learning, a fire detection system was created. These architectures were applied by training models in different modalities, namely thermal and RGB. A multimodal approach to this problem is suggested, where the most accurate thermal and RGB model are combined together in a late fusion fashion.

By implementing several architectures, the evaluation of a single stream network and multimodal architecture became possible. The results clearly indicate that a multimodal approach could increase the performance of the model, compared to traditional CNN approaches. The multimodal architecture achieved a TPR of 100% and FPR of 0% while the best single streamed CNN reached a TPR of 97% and a FPR of 0,02%. The models were evaluated on a testset composed of 300 images taken from a surveying drone, that is independent from the train- and validationset. However, the combination of thermal and RGB data from the exact same scene is too limited to make any definitive conclusions of the multimodal architecture compared to the single stream network. Hence, the amount of varying scenarios in the multimodal testset is less compared to the RGB testset. During the project, extensive data augmentation of the original dataset was made and compared to the original dataset, which led to a conclusion that augmenting data could result in an increase in performance.

convolutional neural network, multimodal, late fusion, deep learning, CNN, fire detection, wildfire, transfer learning

Acknowledgements

We would like to thank Amir Shahroudy for his invaluable ideas and confidence throughout this project. We also want to thank Fredrik Kahl for kindly helping us whenever we needed answers. We also want to show our gratitude towards Pulkit Chugh for his guidance in the beginning of the project. A special thanks to Andreas Björnberg at Carmenta, for his confidence in this project and in us. Last but certainly not least, we would like to thank Valencian Agency for Security and Emergency Response (AVSRE), who shared their data of wildfires, without them the thesis would not have been possible.

Daniel Posch, Jesper Rask, Gothenburg, May 2019

Contents

List of Figures

1
Introduction

The forests are one of the key characters in nature, they stabilize soil, store water, dictate weather patterns and work as the planet's lungs by manipulating the carbon dioxide levels in the atmosphere [3]. The forests also have an economic impact, due to its contribution to a large number of job opportunities. Unfortunately, there has been a significant increase of wildfires in the past years. Since 2000, an average of 73 000 wildfires ravages 7 million acres every year [17]. This threatens both human infrastructure as well as the ecosystem near the fires. Therefore, forest fire fighting is one of the most important methods in the preservation of natural resources. Traditional methods of forest fire monitoring include watchtowers near vulnerable areas, air patrols and calls from the general public. These are inefficient methods considering vulnerable areas could change from year to year, and requires human interaction which could potentially threaten the observers' safety.

In the age when we witness a long-term rise in the average temperature of the planet, any reduction of CO_2 emissions is critical to prevent the irreversible increase of the world temperature. Although estimates vary and have some uncertainties, experts approximate that the wildfires account for up to 20% of the greenhouse gas emissions, and are estimated to increase [20]. Furthermore, wildfires and global warming are two phenomena that reinforce each other. Consequently, prevention and suppression of wildfires is a growing task. Carmenta Public Safety [29] focuses on emergency response systems with customers all over the world which handle call taking, dispatch and communications technologies. Customers of Carmenta, who experienced an increase in the number of wildfires in recent years have expressed the need for reliable fire detection technology in order to aid firefighters with valuable information such as fast localization and area affected. The company wants an implementation of a fire detection system that can be used by an unmanned aerial vehicle (UAV), equipped with different sensors in order to safely extract information of the wildfire. A UAV with computer vision techniques is capable of covering the need for a safe, mobile and efficient wildfire monitoring.

1.1 Aim

This thesis aim to develop a computer vision based technique suitable to achieve a fast and robust fire detection system from the perspective of a UAV. The system will be able to locate if there is a fire present in an image and locate where in the

image the fire is. When a fire is located, its coordinates are exported to the fire visualizer and displayed as an affected area. The solution will be based on both RGB and thermal images in an attempt to increase the robustness of the system. Furthermore, we will investigate if one could benefit from using both RGB and thermal images as input to a two-streamed neural network to detect a fire in a frame which, as a result increases the true-positive and true-negative rates.

By implementing a two-streamed Convolutional Neural Network (CNN) based on the leading architectures in the field, the objective is to achieve a binary classification task with high accuracy compared to previous models and yet maintain a speed that is viable for a real-time system. The goal is to produce one CNN for each modality and combine them as a multimodal network. The reasoning behind creating a classifier and not a localization network is that the precision gained from a localization system is not needed for mapping an area of a fire. This is due to the fact that the fire department want information about the surrounding environment such as vegetation, buildings, etc. of the fire. Hence, detailed bounding boxes are not needed, minor segmentation of the image is adequate.

The main questions that this thesis will address are:

- Could a two-streamed multimodal CNN's binary classifier outperform previous approaches that are explicitly based on color values and temporal information in a fire in terms of detection rates?
- Could data augmentation on the relatively small dataset increase the performance of the classification model?
- How does a two-streamed multimodal CNN perform compared to using a single streamed CNN in terms of fire detection?
- Is this approach appropriate for the hardware equipped on a UAV?

1.2 Approach

To answer the presented research questions, the work is divided into three parts. The first step is to create two separate fire detection algorithms, one classifying RGB images and one thermal images. The second step is to combine these two CNN's in a multimodal fashion, by using both thermal and RGB images as input to the CNN. The final step of this thesis will evaluate the different solutions in terms of precision and speed to conclude if this solution is suitable for a real-time drone-based fire detection application.

1.3 Delimitation

This project will develop and evaluate a computer vision system, that could potentially be equipped on a UAV. The data provided from Carmenta is from real-life scenario video streams of wildfires, taken from sensors that were placed on a manually driven UAV. The videos extracted are both from sensors of an RGB camera

and a thermal camera. Therefore, this project will touch neither the hardware nor the control system that is needed for maneuvering the UAV. The UAV perspective will be simulated using the data provided by Carmenta.

We will take into account that a UAV-based machine has limited processing power, but will assume that the vehicle is equipped with the highest level of general development hardware.

The system will extract potential candidates of fire exposed regions and if the candidate is evaluated to be a real fire, that region should be plotted on the map. The map API is a developed software provided by Carmenta, and will be used for the creation of the visualization application.

1.4 Outline

In this thesis, the theory behind neural networks will be explained, more precisely the different techniques required to build a convolutional neural network, what the different layers do and why they are used. Furthermore, different techniques used to prevent low accuracy of a neural network model will be covered. The thesis will illustrate different multimodal approaches to deep learning such as early, middle and late fusion.

In the methodology section, the different frameworks used during the project will be presented. The experiments conducted on presented network structures will be covered, as well as how the multimodal approach was solved. Furthermore, techniques regarding training a neural network such as training a CNN from scratch and using a pre-trained models are evaluated. The construction of a multimodal model and how it is trained are introduced, followed by an explanation of the evaluation process. The training process is further explained by showing the structure of the dataset, the pre-processing steps and the data augmentation performed on the dataset. Finally, the implementation of the CarmentaEngine application is presented.

The result chapter will cover the experimental outcome of our approach compared to benchmarks presented by the baseline models using the evaluation metrics explained in the methodology chapter. Lastly, future work and the results of the thesis are discussed in the discussion and conclusion chapters.

2
Theory

2.1 Artificial Neural Network

Since the discovery in 2010 of the computational power in GPUs (Graphical Processing Unit) for machine learning [31], the field of neural network has massively grown in popularity. This section aims to simplify the idea of the biological model of neural networks and explain how artificial neural networks grew from that idea. Furthermore, this section will introduce the fundamentals of convolution neural networks and its vital parts.

2.1.1 Biological structure

By looking at an image, the human brain can extract a huge amount of information in seconds. Information about the scene, people's identities, actions and even intentions. The human brain is capable of unfolding an image of RGB values into information, and so are computers.

To replicate the function of the human brain, one has to understand the structure of the nervous system. The neuron doctrine Cajal [5] state that the brain is made from elementary signaling units called neurons. There exist many different neurons, but they all share some basic components.

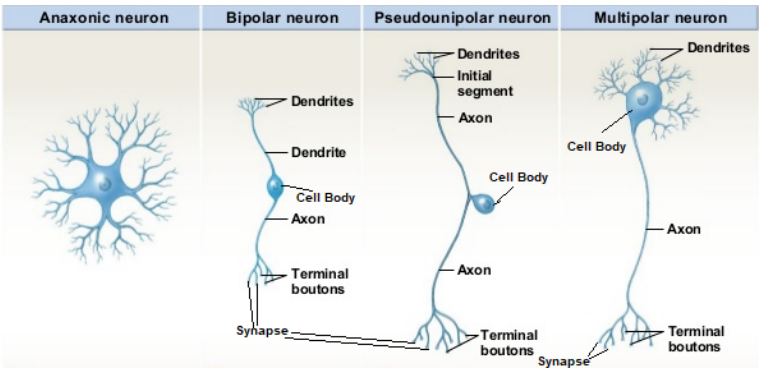


Figure 2.1: Examples of different neurons sharing the same components [8].

As shown in Figure 2.1, the basic morphology of the neurons are the axons, dendrites, cell bodies and synapses. One analogy is to think of a neuron as a tree where the dendrites, axon and cell body represent the branches, root and trunk of the tree [44]. Information flows from one neuron to another across a synapse, which is the contact

point between them. The dendrites form a fine filamentary bush with thinner fibers than the axon and work as the input channel for a neuron. The axon is a thin cylinder carrying impulses from the body to other cells and splits into endbulbs as the output channel. A neuron could have several dendrites, however, only one axon [44].

The given description of a neuron is heavily simplified, compared to how the neuron actually operates. The individual neurons are complicated, as there exist hundreds of different classes of neurons, they have a myriad of parts, control mechanisms and sub-systems. Together, the different types of neurons form a process which is asynchronous and not continuous. The modern artificial neuron try to replicate the most basic functions of this highly complex organism.

2.1.2 Artificial neurons

The artificial neuron is the primary unit of a neural network, that simulates the most basic functions of the biological brain neurons.

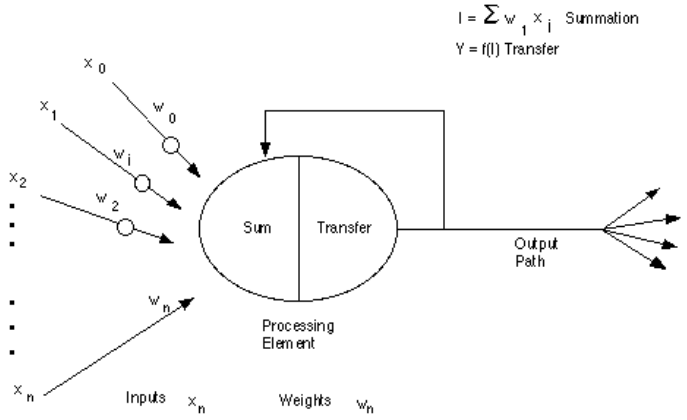


Figure 2.2: An artificial neuron with basic functionality [33].

The artificial neuron represented in Figure 2.2 receives multiple inputs represented by X_n that is multiplied with a connection weight W_n. The products are summed and fed through a transfer function and produce an output. This is one example of how an artificial neuron operates and it is not the general structure, since there exist several implementations of structures that utilize different summing and transfer functions.

The output of the summing function is sent as input to the transfer function. The transfer function takes the input and transforms it to a certain value depending on the function. The most common transfer functions are Sigmoid, Tanh, ReLU and Softmax.

ReLU function The Rectified Linear Unit is most commonly found in deep learning models and is defined as the positive part of the argument x:

$$f(x) = x^+ = max(0, x),$$

where x defines the input to a neuron. The function returns a non-negative value x > 0, otherwise it returns 0, see Figure 2.3c on the following page.

Sigmoid function The logistic sigmoid function is mathematically defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$

The function is characterized by its S-shaped curve as can be seen in 2.3b on the next page. The return value of the function tends towards zero when the input moves towards negative infinity while the function tends to one when the input moves towards positive infinity. The sigmoid function is often used as the output layer's activation function for binary classification problems [33].

Tanh function The hyperbolic tangent function is a rescaled version of the sigmoid function, but the output values ranges from [-1,1]. Strongly negative inputs are mapped to negative outputs, inputs with the value zero will be mapped close to zero and large positive values are mapped to a positive output [48]. The mathematical expression is defined as:

$$\tanh(x) = \frac{\sin(x)}{\cos(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

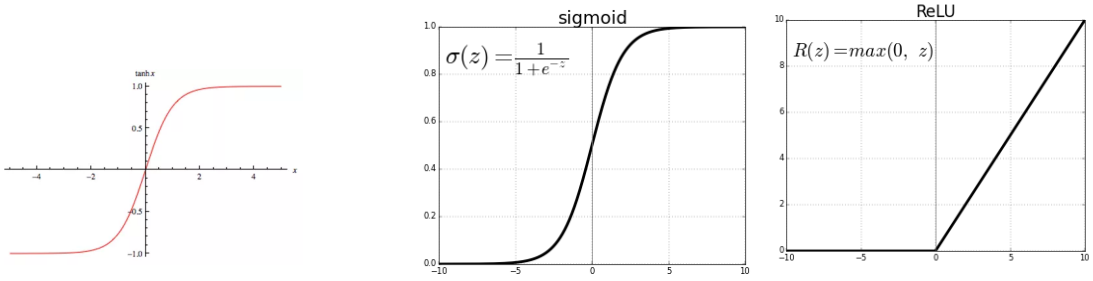
and visualized in Figure 2.3a on the following page.

Softmax function The softmax function is an exponential activation function that is commonly used in the output layer, where the input values $S(y_i)$ are normalized into a vector that is fed into a probability distribution whose total sums up to 1 with range [0,1]. Since the softmax function has a range [0,1], it is suitable to apply the softmax function in network with multiple classes [15]. The mathematical expression of the softmax function is defined as:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}.$$

2.1.3 Artificial Neural Networks

A neural network consist of groups of artificially created neurons, called layers. An artificial neural network typically consist of an input layer, a number of hidden layers and an output layer, as visualized in Figure 2.4 on the next page. Each layer consists of multiple artificial neurons, and the information that flows between the



(a) Tanh **(b)** Sigmoid **(c)** ReLu

Figure 2.3: A graphical representation of the transfer functions described [48].

neurons is indicated with the black lines.

The first layer presented in Figure 2.4, is the input layer. This layer is considered to be passive, meaning that it does not change the data. The input layer's neurons receive values on their input channel and passes that information to their individual connections. Unlike the input layer, the hidden layers are considered to be active layers, meaning they can modify incoming data. In Figure 2.4, each value is sent to all the hidden neurons (represented by the arrows), this is called a fully interconnected structure. The number of hidden layers differ for every network and are heavily dependent on what type of problem the network is trying to solve. The hidden layers may also use different types of transfer functions, such as ReLU, Tanh and Sigmoid. The problem in question and what data is being processed are two deciding factors when it comes to choosing the right transfer function [45].

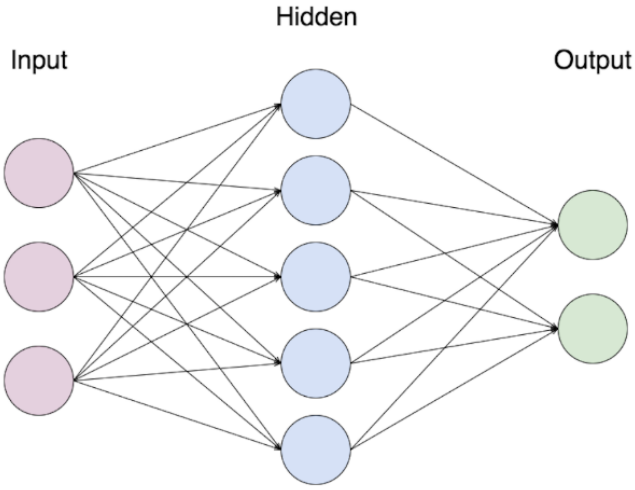


Figure 2.4: An example of a three layered neural network, compromised of an input layer, a hidden layer and an output layer [42].

2.1.4 Convolutional neural network

Deep-learning networks are commonly distinguished from the shallow neural networks by their depth. A neural network that consists of more than one hidden layer is generally defined as a deep neural network. The convolutional neural networks are similar to the neural network as they are made of neurons with learnable weights and biases, and the whole network still expresses a single differentiable score function. However, the convolutional neural network is typically defined by the type of hidden layers it uses, such as convolution, fully connected, normalization and pooling layers.

Using neural networks for image pattern recognition works well for small images that are single layered and consists of simple shapes. However, using fully-connected neural networks on images with high resolution and three color channels would lead to a huge number of trainable parameters. As a result of the increased amount of parameters, the model will probably prone to overfit.

Convolutional neural network has its neurons arranged in a 3-dimensional structure: width, height and depth. Rather than focus on one single pixel at a time, the convolutional network uses square patches of pixels and passes them through a filter, called a kernel. The purpose of the process is to transform the image into a form that is easier to process, without missing any features that are critical for the prediction.

2.1.5 Convolution

The convolution operation is the main building block of a Convolutional neural network. The convolution takes a filter or kernel of a specified size and slides it over the input with a given stride, that specifies how many columns the filter should move on the input image. A dot product between the filter and section of the input (the same size as the filter) is computed in each step, shown in Figure 2.5 on the following page. The output generated from each step are summed into a feature map which is put together as a final output [9]. In purely mathematical terms, the operation is defined as a linear operator that transforms data from one domain to another:

$$(f \cdot g(t)) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau. \tag{2.1}$$

The input image I is on the left side of Figure 2.5 and the filter K is in the middle. Due to the shape of the filter, it is annotated as a 3x3 convolution. To produce the feature map, element-wise matrix multiplication is applied at every location, and the result is the sum, annotated as I * K on the right side of the Figure.

2.1.6 Pooling

Pooling, or down-sampling is used to reduce the dimensionality of feature maps. Pooling with a 2x2 feature map, for example, takes four pixel values as input and output a value for those pixels. As shown in Figure 2.6a, 2x2 pooling on an 8x8 image will result in 16 feature maps presented in a 4x4 matrix. There are different types of

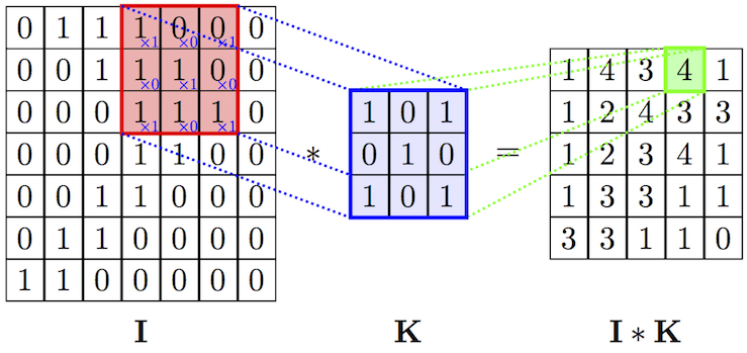
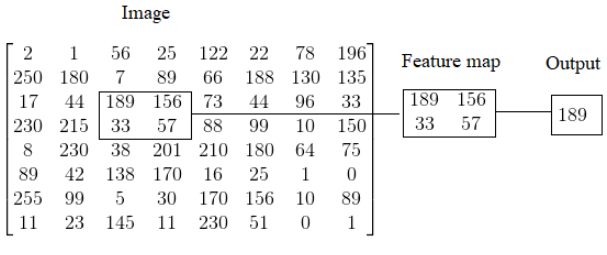


Figure 2.5: An example of a convolution where I is the input, K the kernel and I * K is the output [24].

pooling such as average, mean, max, min and stochastic pooling. The pooling types are distinguished by how they choose pixel values to create their feature maps. The less obvious of the pooling types; stochastic pooling picks a value randomly, where high pixel values have a higher chance to be picked [1].



(a) 2x2 Max-Pooling example

$$\begin{bmatrix} 250 & 89 & 188 & 196 \\ 230 & 189 & 99 & 150 \\ 230 & 201 & 210 & 75 \\ 255 & 145 & 230 & 89 \end{bmatrix}$$

(b) 2x2 Max-Pooling result

Figure 2.6: (a) Example of a 2x2 max-pooling with stride = 2, where (b) is the resulting output.

2.1.7 Prevent overfitting

Overfitted models is one of the most common problems researchers and companies encounter in the field of deep neural networks. One of the reasons is that many state of the art architectures have a large number of parameters to be learned during training. Overfitting occurs when a trained model has a high validation accuracy on the training data but fails to perform well on test data. More specifically, the trained model learns the behavior of the noise patterns in the training data which creates a large difference between the training and test error, which is the definition of overfitting [34] and is visualized in Figure 2.7.

Deep neural networks generally have a high amount of trainable parameters. A complex model that contains more parameters than can be justified by the amount of training data, leads to an overfitted model. This is one of the major banes of big data, and its relevance is growing. In Figure 2.7, the model fit the data perfectly during training, however, in samples outside the train data it performs badly. To

avoid overfitting, one could either increase the amount of training data or enhance the networks ability to generalize by applying different techniques explained below.

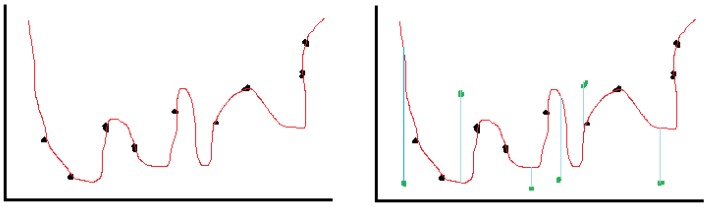


Figure 2.7: Left - a figure where the line intersects the black dots perfectly(overfitted), and the right image when the model performs poorly on the test data(green dots), where the error is huge(blue lines).

Regularization is a technique used in order to prevent overfitting. The most commonly used regularization algorithms are L1, L2 and dropout. The L1 and L2 regularization updates the cost function by adding a regularization term which decreases the values of the weight matrices. The assumption is that a network with smaller weights matrices leads to simpler models. The difference between L2 and L1 regularization is that L2 decay the weights towards zero, but not exactly zero, while L1 regularization could be reduced to zero [35].

$Costfunction(L2) = Loss + \frac{\lambda}{2m} * \sum ||w||^2$.

The L1 regularization penalize the absolute value of the weights, meaning that a neuron with a high weight will cost more than a neuron with a low weight, therefore L1 could be useful while trying to compress the model [35].

$Costfunction(L1) = Loss + \frac{\lambda}{2m} * \sum ||w||$, where w = each weight in the neural network.

Dropout is a commonly used technique to enhance the network's ability to generalize the data. The method randomly drop neurons in the neural network and temporarily removes the incoming and outgoing connections. The dropped neurons contribution to the activation of downstream neurons are temporally removed. Neighbouring units will have to compensate for the removed unit and handle that specific representation of the missing neuron. The effect of the dropout is that the network becomes less dependent of the specific weighted neurons and therefore enhance the networks ability to generalize [39].

2.1.8 Loss

When optimizing an algorithm, a function is used to evaluate a candidate solution, also called an objective function. By applying an objective function such as cost and

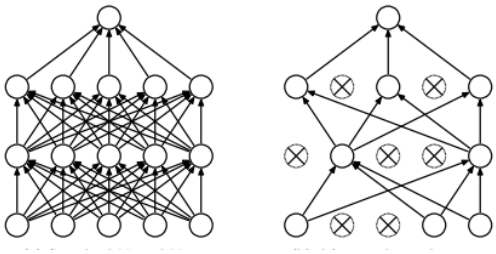


Figure 2.8: This figure shows the effect of dropout. The left network is the original neural network. The right network is the result after applying dropout [39].

loss functions one seek to maximize or minimize the objective function. Normally in deep learning, it is preferred to minimize the objective function which calculate a possible solution, in this case a set of weights with the lowest cost. The value returned by the loss function is called "loss" [4]. The loss is a measurement that represent the classification models performance. For example, a model that predict a probability of 0.1 when the observation labels value is 1, would result in a high loss.

Some of the functions that are commonly used as loss functions are cross-entropy and mean square error. When calculating the cross-entropy, one seeks the set of the model weights that minimize the difference between the model's predicted probability distribution given the dataset and its distribution of probabilities [4]. Depending on the task of the neural network, binary cross-entropy or categorical cross-entropy can be applied. If the network is trying to classify multiple classes from the dataset, applying categorical cross-entropy is sufficient. If the dataset contains only two classes one can apply binary cross-entropy that predicts the probability of the test dataset belonging to one class [4].

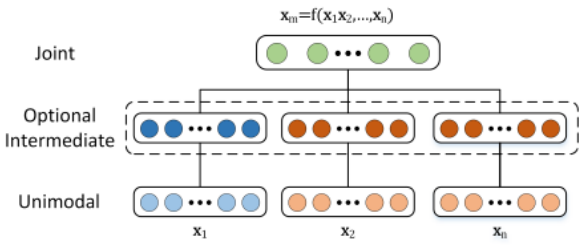
2.2 Multimodal deep learning

The world is interpreted in a multimodal way, there are sounds, objects, flavors and many more modalities. A modality refers to the way something is experienced or categorized as. In order to create an artificial intelligence that is able to interpret the world around us, it is sufficient to provide it with a sufficient amount of information, from different sensors.

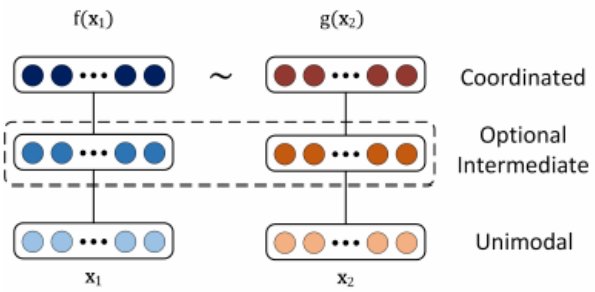
The field of multimodal machine learning offers the possibility to capture correlations between the modalities, but also brings a source of difficulty. To create a successful machine learning model, one need to provide the model with appropriate data. The challenge of representing raw data in a way that is understandable for a computer has always been a challenge in the machine learning industry.

The representation of different modalities could be done by combining the unimodal representations into a multimodal space, called joint representation. One example of this method is early fusion, using concatenation of the individual features. Another

approach which is called coordinated representation, learn the representations of each modality and coordinate them through specified rules e.g. Euclidean distance. One example of a coordinated representation builds on the distance between the different modalities in the coordinated space. Hence, the distance between different modalities are estimated as the ground truth. For example, an image of a fire and the word "fire" have specific distance d1 between them. The word "notFire" and an image of a fire also have a distance d2 between them. In this example, the distance d1 should be less than d2. A Figure of the two approaches is shown in Figure 2.9



(a) A representation of the joint model.



(b) A representation of the coordinated model.

Figure 2.9: This figure shows the joint (a) and the coordinated (b) representation. In figure a, the unimodal representations are combined in a joint multimodal space. The coordinated representation operates in their own space, but are coordinated through rules [2].

One part of the multimodal machine learning relies on how to translate data from one modality to another. The approaches to translation are very broad and therefore it is possible to categorize them into two sub-problems, example-based and generative translation. **Example-based translation** refers to the problem of when a dictionary is used to translate the information between modalities. **Generative translation** however, is supposed to generate a translation such as symbols [2].

The fusion stage of a multimodal approach refers to the problem of integrating information from one modality to another modality. This allows the network to interpret a situation based on different modalities. By having access to information from multiple modalities of the same phenomenon, the model generally produces more robust

predictions. Furthermore, if one of the modalities does not operate correctly, the model can still rely on the individual modality. There are several ways of integrating the information in a model. In literature, three main approaches for the fusion process of a multimodal network, that is the decision-based fusion, feature-based and hybrid multi-level fusion.

The Early fusion approach, also called feature-based fusion, is applied at the feature level where the different modalities are appended. After the extraction of features, they are immediately integrated. This results in a single model where both the parameters and the classification task is optimized for all modalities applied [12].

The late fusion is known as decision-based fusion and applies the integration after each modality has made a decision. When applying late fusion, the idea is to train the different models separately, when a satisfying result is achieved, one discards the output layer and apply fully connected layers and a joint output classification layer. Once the fusion is complete, the joint models are trained to optimize the joint parameters [13].

The hybrid fusion approach combines and tries to extract the advantages of the early and late fusion approach. The hybrid fusion integrates features both along the recognition and at the decision levels [2].

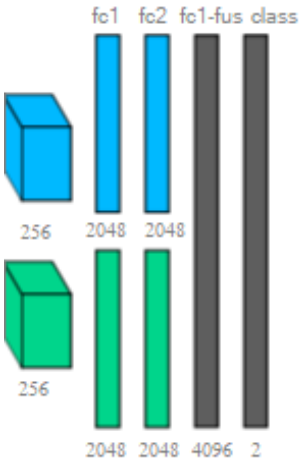


Figure 2.10: An example of late fusion between two networks (blue and green) with a fully connected layer and a binary output (black) [13].

2.3 Related work

The traditional approaches in the area of fire detection are purely based on the color of the frame, or a combination of the color and temporal information in image sequences. Some of the algorithms that have been proposed in the last two

decades utilize an evaluation of the visual spectrum and apply the different variations of the color space, YC_bC_y [47], L*a*b* [6], HSV [7], RGB [32] and combined models of color spaces [28]. Some of the existing vision-based systems detect fire based on both color and motion features and try to increase the accuracy of their system by combining features of a fire [47] [19]. By combining features of fire and smoke, the different approaches get an acceptable accuracy in a specified dataset. However, the algorithms tend to have poor robustness when using a different testset.

2016, Cruz et al. [10] implemented two indices, one Fire Detection Index(FDI) and one extension of it called Forest Fire Detection Index(FFDI), which was developed by subtracting the green and blue from the red in an RGB image and using a parameter ρ as a weight factor:

$$\rho(2r - g - b) - (2g - r - b).$$

The traditional approaches are dependent on the dynamic features of a fire, thresholds and motion features. These dependencies could be seen as restrictions in many cases. Henceforth, the most recent approaches in this field are transitioning to a convolutional neural network approach, which has shown great results in the field of image classification.

AlexNet [16] is comprised of eight layers with weights. The network has five convolutional layers which are followed by three fully connected layers. The output of the two first convolutional layer is followed by a max pooling layer and normalization.

InceptionV3 [36] has 22 layers (considering an inception module as one layer) and is much faster than AlexNet and yet more accurate. One concept that characterize InceptionV3 is that they make use of inception modules. The idea was to cover a larger area while maintaining a detailed resolution, by convolving different sizes of the most accurate detailing. A straight forward approach of how to improve the accuracy in deep learning, one could increase the number of layers. InceptionV3 uses 9 inception modules, which offers superior performance compared to AlexNet. However, due to the increased number of layers, there is a higher chance for the model to overfit.

VGG-16 [37] architecture consists of 16 layers, five convolutional blocks(conv blocks) and ends with a fully connected classifier. The first two conv block has two 2D conv layers and a 2D max pooling layer while the three later conv blocks consist of three 2D conv layers and ends with a 2D max pooling layer. The fully connected classifier flatten the array and then apply three dense layers to construct the output of the network.

DenseNet [18] The DenseNet consists of a set of Dense blocks where each pair are separated by a transition layer. A dense block consists of a batch normalization, ReLu activation and a 3x3 conv layer. Every transition layer is made out of one batch normalization, one 1x1 conv layer and an average pooling which are designed to perform downsampling of the features. The different variants of DenseNet all share the same base components(four Dense blocks and three transition layers).

Dunnings and Breckon [11] proposed an experimentally defined approach to fire pixel region detection within real-time called fireNet, by reducing the number of parameters used for the network. The reduction of parameters could offer an increase in speed of 3-4 times compared to the parent network (Alexnet) and a detection rate of 93%.

Zhao et al. [50] took the approach of combining a saliency detection based strategy, to extract the ROI which is then used as an input to a CNN. The network was evaluated to reach a 98% accuracy rate and could process around 40 fps.

Zhang et al. [49] proposed a faster R-CNN(Region CNN) to detect wildland forest fire smoke. The dataset was produced by both combining real images of smoke and synthetic smoke images, to enhance the performance. In a field that is not as common, it could be difficult to acquire the data needed to train a CNN to recognize the feature(s). To prevent a model from over- or underfitting researchers have applied different pre-processing techniques. Namozov et al. [26] proposed a novel CNN for both fire and smoke detection, with a limited dataset. Instead of using traditional rectified linear units as the linear units, they used adaptive piecewise linear units to overcome the overfitting problem.

In an attempt to increase the accuracy of CNNs, researchers used multimodal CNNs where multiple inputs was used [13] [46]. The use of multiple inputs let the network learn parameters from more than one modality, which could lead to higher accuracy, since a combination of modalities are better at interpreting a certain situation than an individual modality. Moreover, this would probably lead to a more robust solution, because one modality could still operate if one would fail. Eitel et al.[13] proposed a novel multimodal CNN approach which uses two CNNs, one that takes an RGB image as input, and one that takes an RGB-D image as input. The authors used CaffeNet that starts with five convolution layers and ends with two fully connected layers. To combine the two CNNs Eitel et al. fed both CNNs to a fully connected fusion layer which then is fed to a classification layer which is the output of the network. A similar approach to Eitel et al. was proposed by Wagner et al. [46] who constructed a multimodal CNN using thermal images as well as RGB images for pedestrian detection.

Salman et al. [34] showed that "the increase in loss on the validation set is due to a continuous update of the weights and biases, causing the magnitudes of the inputs of the softmax layer to increase". In the paper they present a novel approach where they increase the validation accuracy by combining different models

and introduce a parameter threshold, however, similar results have been achieved by applying dropout to the combined model compared to the method of introducing a parameter threshold [34].

The mentioned approaches of wildfire and smoke detection are hard to compare, due to the fact that there exists no reference dataset with the right quality nor size. Consequently, the effectiveness of the presented approaches remains ambiguous.

3
Methodology

This chapter consists of the methodological details regarding the conducted experiments and present the deep learning framework used, the datasets and the data pre-processing. Moreover, a description of how the multimodal approach was conducted is given, both regarding the training phase and the fusion of the networks. This chapter will also contain a description of the map integration with CarmentaEngine. The map integration is divided into several parts, that involves transferring pixel positions to real coordinates, a visual representation of the affected area and an update of the drone's position in real-time.

3.1 Frameworks

The machine learning paradigm is something that has emerged during recent years. The frameworks are shifting towards developing models that run on mobile to enhance the flexibility of the application. Today, there exists a myriad of deep learning frameworks at our disposal, that offer an increased level of abstraction along with multiple simplifications. The most important factors of a deep learning framework are performance, flexibility and documentation abstraction. Moreover, one key factor is user activity, which helps when encountering problems with the framework.

Based on the factors mentioned above for the most popular deep learning frameworks, PyTorch [30], Caffe [21], TensorFlow [43] and Keras [22] were considered. For this project, Keras with TensorFlow as backend was chosen. One can choose any neural network framework to implement the same network model that is used in this project, Keras and TensorFlow is just a preferred choice since it has a well-documented API and an active community.

3.2 Fire dataset

A dataset could be structured in many different ways, however using Keras with TensorFlow one would need to structure the dataset in a way as shown in Figure A.1 located in Appendix A.1. An important part of building a dataset is to have a balanced dataset, that is, each class containing the same amount of data for each part in: train, validation and test. If the dataset is unbalanced, the model risk to predict in favour of the class containing the most images.

In order to evaluate the accuracy of the model and allow fine-tuning of the parameters, the dataset was splitted into three different subsets: train, validation and test. The train dataset resulted in 80% of the original dataset, the validation dataset 15% and the test dataset 5%.

3.2.1 Pre-processing

To build an effective classifier, the data has to be in a proper format. In deep learning, usually one consider the input data parameters: number of channels, image dimensions and the range of pixel values. An RGB image typically has three channels corresponding to the color channels red, green and blue, with each pixel is in the range of [0,255]. Our dataset is gathered from series of different sources and the image samples vary in both structure and dimension. Feeding the network with raw data may lead to numerical overflow and might therefore not be compatible with some of the activation functions. Hence, the data needs to be prepared before the training process.

The architecture of VGG16, FireNet, InceptionV3 and AlexNet require the input image to have a square ratio, therefore it was ensured that all the samples had the same aspect ratio. Improper samples were cropped by selecting the center part of the image. Moreover, the dimensions of the image have to be uniform. Therefore, the the built-in function for scaling each image to an appropriate size was used.

Data normalization is applied to ensure that each input parameter to the network has similar data distribution. This was done by subtracting the mean from each pixel and divide the result by the deviation. The normalized data is in the range of [0,1] in our case.

Data augmentation

By performing data augmentation (eg. rotation, zooming, cropping and applying noise such as salt and pepper) to the images in the original dataset, one can create a larger dataset to train a CNN model. The reasoning behind performing data augmentation was to prevent overfitting and therefore increase the performance of the detection algorithm. Therefore, various experiments were conducted with general augmentation techniques of the dataset to evaluate the increase or decrease in performance. After experimenting, it was found that the augmentation techniques that resulted in the highest accuracy improvements are listed in table 3.1.

3.2.2 Dataset construction

Two different pairs of datasets were created for experimenting with both the thermal model and the RGB model. The first dataset resulted in 20 000 RGB images, that were collected from different sources of the internet, earlier fire detection approaches and Carmenta customers. The images were processed as described in Section 3.2.1.

Rotation	40°
Width shift	0.2 pixel
Height shift	0.2 pixel
Sheer mapping	0.2 pixel
Zoom	0-20%
Flip	Horizontally
Fill	Nearest

Table 3.1: Data augmentation parameters. All the values presented in the table are the max-values. The augmentation was executed randomly on every image, ranging from 0 to max-value.

The thermal images were provided by a Carmenta customer, Valencian Agency for Security and Emergency Response and consisted of nine videos of aerial sequences captured from the perspective of a UAV, this resulted in 2500 images.

The second pair resulted in a modification of the original dataset. Extensive data augmentation on the original datasets was performed, to reduce the risk of overfitting. The RGB dataset ended up having 70 000 images, and the thermal dataset having 15 000 images. At the end of this procedure, four different datasets were obtained: original RGB dataset, original thermal dataset, augmented RGB dataset and augmented thermal dataset.

3.3 Network structure

We considered several network architectures for this project, with reference to the number of parameters and general performance. Almost all CNN architectures follow the same design patterns of successively applying convolutional layers, downsampling the dimensions while increasing the number of features. Hence, our experimental approach investigated different architectures for each modality.

3.3.1 Network candidates

To decide what network architecture to use for the the fire detection task, several architectures have been implemented and evaluated. Two major factors that influenced which networks to further examine, was accuracy and the time elapsed for each image to be classified. Four network architectures were used as a base for this experiment, all with minor modifications. The networks that were used as the core structures were VGG16, InceptionV3, DenseNet and FireNet. These were modified as binary classifiers, trained and evaluated.

InceptionV3 One of the larger networks, InceptionV3 was created by Szegedy et al. [41] from Google. The biggest difference between InceptionV3 and other network architectures is the inception modules. The goal of the inception modules is to reduce the parameters without compromising the efficiency of the network. The

different inception modules A, B and C are located in Appendix A.2 on page II.

InceptionV3 modified			
Nr	Layer	Tensor size	Parameters
1	Conv(3x3, stride 2)	99x99x32	0.8k
2	Conv(3x3, stride 1)	97x97x32	9.2
3	Conv(3x3, stride 2) padded	97x97x64	18.4k
4	Pool(3x3, stride 2)	48x48x64	5k
5	Conv(3x3, stride 1)	48x48x80	138k
6	Conv(3x3, stride 2)	46x46x192	12k
7	Conv(3x3, stride 2)	46x46x192	9.2k
8	Conv(3x3, stride 1)	22x22x64	55.3k
9-12	3xInception Module A	22x22x288	
13-17	4xInception Module B	10x10x768	
18-20	2xInception Module C	4x4x1280	
21	Global Avg. Pooling	1x1x1024	
22	Dense	1x1x1024	2,1 M
23	Softmax(Output)	1x1x2	2k

Table 3.2: Detailed description of the InceptionV3 architecture [41]. For simplicity, each inception module is referred to as one layer. All conv. layers and inception blocks use 0 padding.

VGG16 VGG16 belong to the "historical" convolutional neural network models, proposed by Simonyan and Zisserman [37]. The model achieved 92.7% accuracy in ImageNet [25], which gave the model a top-5 accuracy score. The input image is passed through a series of convolutional layers, with filters using a small receptive field of 3x3. Spatial pooling is carried out by the four max-pooling layers that lie after some of the convolutional layers. Moreover, a global average pooling layer is added before the fully-connected layer. Max-pooling is performed using a 2x2 pixel window, with the stride set to 2.

VGG-16 modified			
Nr	Type	Tensor size	Parameters
1	Conv2D(3x3)	224x224x64	1.7k
2	Conv2D(3x3)	224x224x64	36.9k
3	Max-pooling(2x2, stride 2)	112x112x128	-
4	Conv2D(3x3)	112x112x128	73.8k
5	Conv2D(3x3)	112x112x128	147.5k
6	Max-pooling(2x2, stride 2)	56x56x256	-
7	Conv2D(3x3)	56x56x256	295k
8	Conv2D(3x3)	56x56x256	590k
9	Conv2D(3x3)	56x56x256	590k
10	Max-pooling(2x2, stride 2)	14x14x512	-
11	Conv2D(3x3)	14x14x512	1.18M
12	Conv2D(3x3)	14x14x512	2.36M
13	Conv2D(3x3)	14x14x512	2.36M
14	Max-pooling(2x2, stride 2)	7x7x512	-
15	Conv2D(3x3)	7x7x512	2.36M
16	Conv2D(3x3)	7x7x512	2.36M
17	Conv2D(3x3)	7x7x512	2.36M
18	Global Avg. Pooling	1x1x512	-
19	Fully-connected	1x1x1024	52k
20	Softmax	1x1x2	2k

Table 3.3: Detailed description of the VGG-16 architecture. Unless stated otherwise, padding is 0 and stride is 1.

DenseNet-201 A problem arise with CNNs when they are too deep. The result of deep networks is that the information flows through to many layers, leading to information loss. The DenseNet architecture, see Figure 3.1, tries to increase the depth of a CNN, without the information loss. The architecture explicitly differentiates between information that it already has and what is added. The layers are narrow and make use of small sets of feature-maps. The maps are added to the networks collective knowledge, while the remaining maps are unchanged. The Layers in DenseNet are connected in a specific way as shown in Figure 3.1, which requires fewer parameters than the traditional CNN and results in less redundant feature maps [27].

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 3.1: Architecture of different DenseNet architectures.

FireNet invented by Dunnings and Breckon is a modified AlexNet [11]. The idea behind FireNet is to lower the trainable parameters but still maintain accuracy. The authors gained a 3-4x speedup compared to the original AlexNet, while the accuracy was reduced from 92% to 91% [11].

FireNet modified			
Nr	Type	Tensor size	Parameters
1	conv(5x5)	196x196x64	4.8k
2	nonlinearity	196x196x64	-
3	Max-pooling(3x3, stride 3)	65x65x64	-
4	conv(4x4)	62x62x128	131k
5	nonlinearity	62x62x128	-
6	Max-pooling(3x3, stride 2)	30x30x128	-
7	conv(3x3)	28x28x256	295k
8	nonlinearity	28x28x256	-
9	Max-pooling(3x3, stride 2)	13x13x256	-
10	conv(1x1)	13x13x512	131k
11	nonlinearity	13x13x512	-
12	Max-pooling(3x3, stride 2)	6x6x512	-
13	Fully-con(4096)	4096x1x1	75.5M
14	nonlinearity	4096x1x1	-
15	Fully-con(4096)	4096x1x1	16.7M
16	nonlinearity	4096x1x1	-
17	Fully-con(2), Softmax	2x1x1	8k

Table 3.4: Detailed description of the FireNet architecture. Unless stated otherwise, padding is 0 and stride is 1.

3.3.2 Multimodal network

Our network for the multimodal approach resulted in two models of different modalities that were fused together. For simplicity, the two models are denoted as: RGB-model and Thermal-model (see Figure 3.2). The multimodal network constructed requires two datasets, that have the same resolution of the images. One dataset consist of RGB images and the other dataset of thermal images. Two input tensors was created, one for RGB images and one for thermal images which differ in terms of channels, where the RGB tensor has three color channels and the thermal tensor has one. The input tensors are fed to the pre-trained RGB- and Thermal models. In order to make the two models compatible, a fully connected layer(FC1) was added to each model and was then merged. After the merge, a fully connected layer was added and an output layer (Fusion-FC1 and Output) as shown in Figure 3.2.

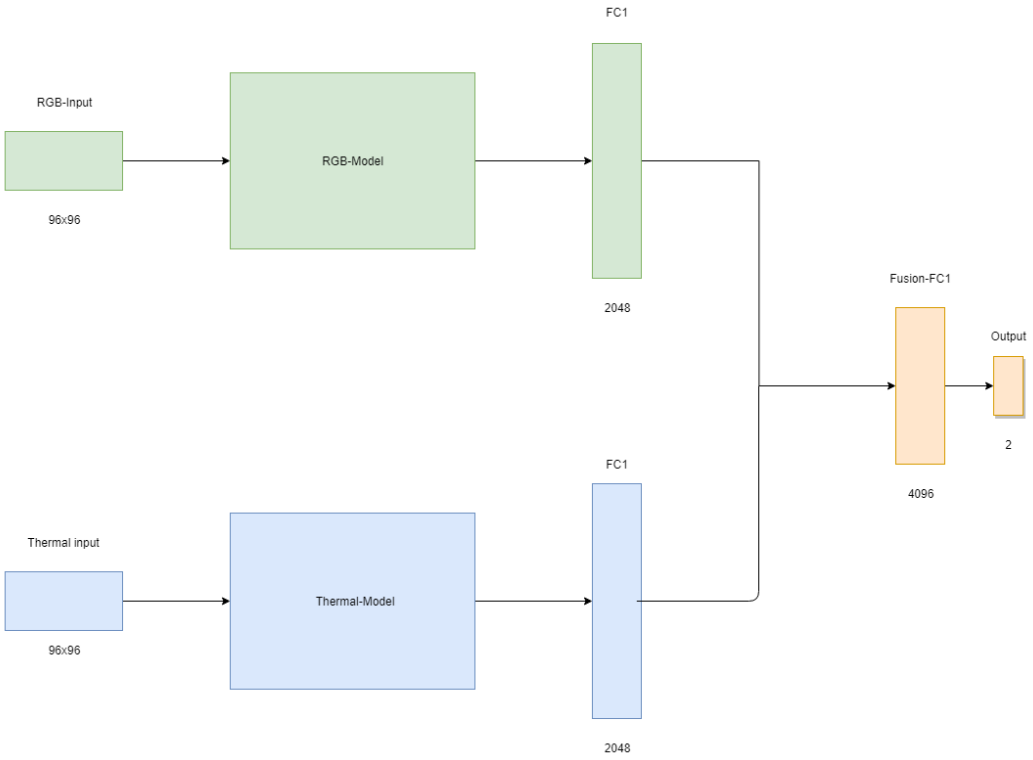


Figure 3.2: The multimodal architecture where the RGB and thermal model are merged in a late fusion fashion.

3.3.3 Training from scratch

By training the mentioned architectures without any initialized weights, it was noticed that the model suffered from overfitting, which was a consequence of training a large model with a small dataset. To prevent overfitting the idea of training a model with uninitialized weights was abandoned and turned to transfer learning, which is described in Section 3.3.4.

3.3.4 Transfer learning

Transfer learning is a technique that is commonly applied when working with limited amount of data. The benefit of transfer learning is that one can acquire a pre-trained model with its weights. By removing the output layer and adding their own desired output one reduce the problem with limited datasets.

In the project, transfer learning was used for VGG16, InceptionV3 and DenseNet where in all three cases the output layer was removed, and a new output layer with two classes (fire, notFire) was added. Before the training process of VGG16 began, all layers was unfrozen in order to make all parameters more optimized toward our dataset.

The transfer learning approach to InceptionV3 was different compared to the VGG16 approach. When implementing InceptionV3 for two classes, all layers up to the last two inception blocks was frozen and only the remaining layers were trained. Then the entire network was unfrozen and trained with the complete dataset, which was proposed in the Keras documentation.

3.3.5 Training multimodal architecture

To acquire the multimodal-model, it was needed to train the models in different steps. First, the weights from imageNet was initialized for the RGB-model. After transfer learning was applied, the RGB-model was trained with all the layers unfrozen. The transfer learning step was done twice, once for the RBG-model and once for the thermal-model using our RBG dataset and thermal dataset. When training of the two models were finished, they were fused together with the added layers described in Section 2.2 about late fusion. When training the combined models (multimodal), all layers were frozen except the newly added layers: FC1 (blue and green), fusion-FC1 and output (black) as can be seen in Figure 3.2. The complete model was trained using our thermal and RGB dataset. However, when training the last layers of the multimodal model the "multimodal dataset" had to be balanced. Therefore the RGB dataset had to be reduced to 15,000 images, since the augmented thermal dataset contained 15,000 images.

System specification All training was done on Intel Core i7-8700 3.2 GHz, 32 GB RAM and a Gigabyte GTX 1080 TI 11 GB.

3.4 Evaluation of a model

To evaluate a model, a testset was created. The testset is supposed to be completely independent of the training and validation dataset. To make sure our testset was independent of the trainset, images from various sources from the internet was gathered and manually made sure the images gathered was different from the rest of the dataset. The testset resulted in 150 non-fire images and 150 fire images. The

image samples varied in terms of camera angle, lighting, distance from the scene and amount of smoke compared to fire in order to cover as many variations of fire scenarios as possible.

To evaluate the predictions of the model, the term accuracy was investigated. Accuracy is the ratio of correct predictions in relation to the number of predicted samples:

$$Accuracy = \frac{NumberOfCorrectPredictions}{TotalNumberOfPredictions}.$$

For detecting wildfires, accuracy is not enough to evaluate the model, since there could be a great imbalance between the input number of each class. Therefore, a more detailed evaluation model was needed. The problem of only evaluating the model exclusively according to the accuracy, arises when a misclassification of fire is expensive. In order to measure the model's complete performance on a specific testset, four types of outcomes were defined: True positive (TP), true negatives (TN), false positives (FP), false negatives (FN). TP and TN refer to the case when the models predicted label corresponds to the true label. FP and FN refer to the case when the predicted label does not match the true label. Hence, a classifier that does not yield any FN or FP, is considered to be an optimal classifier.

In terms of fire detection, two of the mentioned measurements are of great importance. A fire detection algorithm should capture all occurrences of fire. This corresponds to a maximum number of TP that is in relation to the total amount of positive images. Moreover, the FP rate should be as low as possible, since a high FP rate would trigger a visualization of a fire in the CarmentaEngine, which could be costly if a fire department would act on such false information. The FP corresponds to the number of negative images in the testset. The evaluation measurements can be calculated with the formulas:

$$TPR = \frac{TP}{P}.$$

TPR = True Positive Rate and P = Number of predictions made(on TRUE images).

$$FPR = \frac{FP}{P}.$$

FPR = False Positive Rate and P = Number of predictions made(on FALSE images).

To extract these measurements, a confusion matrix was created based on the predictions and the corresponding true labels. In Figure 3.3 the TN score was 146, the TP was 148, FP was 4 and FN was 2. This gives a more complete overview of how the model is performing for each class.

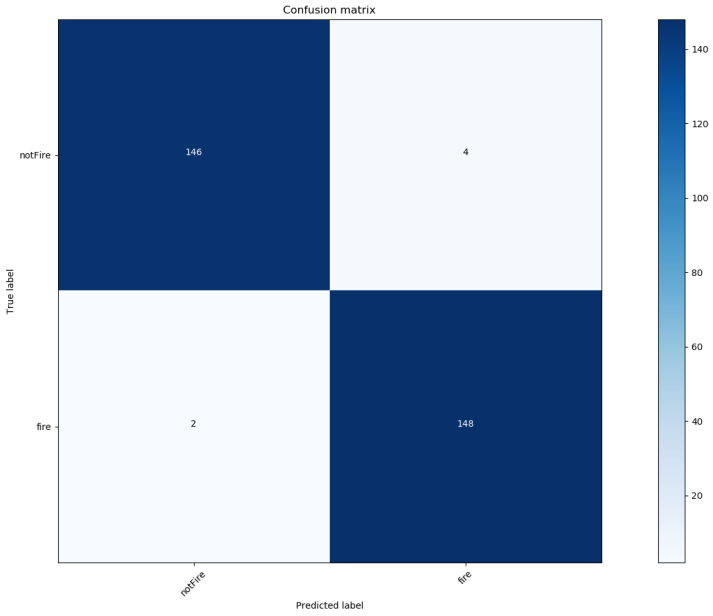


Figure 3.3: Example of a confusion matrix presenting the predictions of the testset that contains 150 non-fire images and 150 fire images.

3.5 Region of interest

There are many different approaches to extract the region of interest, such as labeling bounding boxes for the desired features and let the network train to recognize those features. However, labeling bounding boxes are highly inefficient, since labeling and creating bounding boxes are mostly done manually for each image.

Each image in the testset was split into four equally large segments, in order to get a rough estimate of where in the image the fire is located. It is possible to split the image into even smaller segments, however, there is a trade off between speed and precision of the bounding boxes. In the special case where the fire is located in the entire image, then segmentation of the image would not be needed. Since each segment need to be predicted independently, it was concluded that four segments was enough to get a rough estimate of the area of the fire. The reason being that, using more than four segments would slow down the application without any noticeable gain in area extraction.

3.6 CarmentaEngine application

To visualize the fire detection algorithm's findings, an application that is based on CarmentaEngine, called Fire Visualizer was created. How the Fire Visualizer is connected to the fire detection algorithm is visualized in Figure A.3 on page III. The application is built on a map that is created in Carmenta Studio which visualizes land, water and cities as static objects while dynamic drone objects is updated with its current position on every update of the window. The purpose of the Fire

Visualizer is to graphically display the data of the fire detection algorithm as well as the positions of the drone units. The visualizer allows the user limited interaction with the map and the findings of the drones. The user can interact with the map, such as scaling, panning and by selecting a feature get extended information about the detected fires.

Fire polygons

When a fire has been detected, the application displays the location of the fire by drawing a polygon on the map with the predicted probability printed below the polygon. The polygon is defined by a set of coordinates that follows the world geodetic system 1984 (WGS84LongLat) [14], an ID and the frame that was classified as fire. When hovering over the polygon, the center coordinates of the fire are displayed beneath the polygon and if the polygon is selected, the application opens another window that displays the image that was detected as fire. To keep track of which drone detected the fire, a unique ID for each drone was used. When a fire is found, the ID and position of the drone are used to calculate where the fire is located.

To calculate the area of the fire, the drones current position, FOV and height was extracted and applied to the formula:

$$width = height * Tan(\frac{FOV}{2}),$$

to calculate the hypotenuse by Pythagoras theorem. To get the points in the WGS84LongLat Coordinate Reference System (CRS), the drones current position with the length of the hypotenuse and the azimuth was projected to acquire p1, p2, p3 and p4. When p1, p2, p3 and p4 are determined lines, a line is drawn between them in order to create a polygon to acquire the estimated area of the fire, as shown in Figure 3.4.

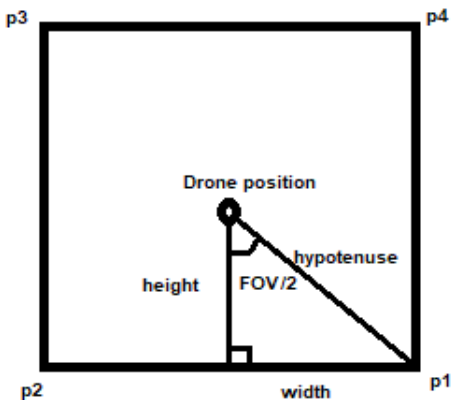


Figure 3.4: Calculations made to acquire the points for drawing the area of the camera view. The drone position, field of view and the UAV altitude is used to calculate the affected area.

4
Results

This chapter describes the results of the experiments conducted during this thesis by first defining how the hyperparameters were defined at the beginning of the project. These parameters play an important role in terms of the optimization and regularization aspects of the training process. The performance of the chosen networks is also presented and compared with the multimodal approach, both in terms of speed and detection rate. Finally, the result of the fire visualizer is presented, integrated with the presented fire detection algorithm.

4.1 Experimental setup

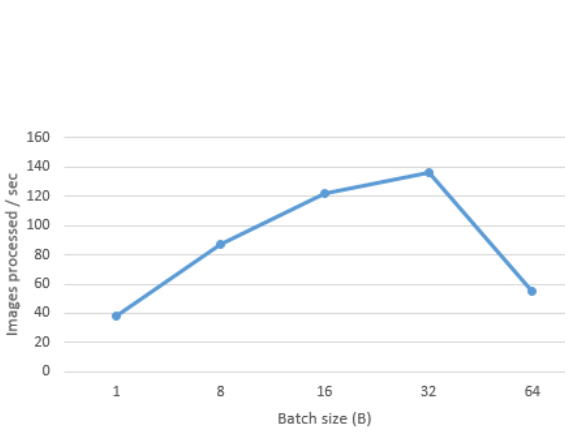
Selecting hyperparameters is an important part of optimizing the model. With strategically chosen hyperparameters, it is possible to enhance the performance of the network in general.

4.1.1 Optimization

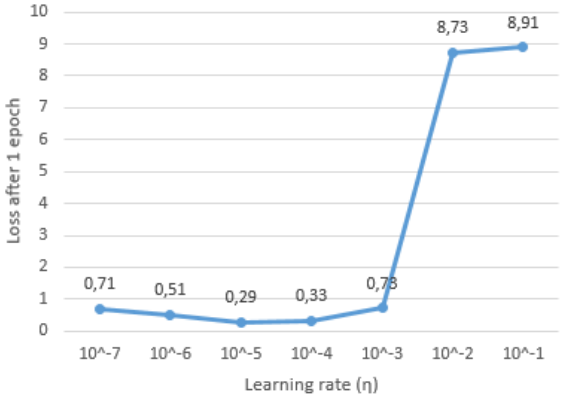
Deep learning models are usually trained by a gradient descent optimizer such as Adam, Adagrad or RMSprop. Adam was chosen as the optimizer, which allows the user to define some of the optimization functions, and the optimal value on these varies between every project. The hyperparameters that were considered to be the most important ones were the momentum coefficient μ, the batch size B and the learning rate η. The value of the momentum coefficient has a default value of $\mu = 0.9$, which has shown to work well in practice for similar projects [40] and was thus fixed at this value throughout the whole project.

The batch size B was chosen with respect to the best computational efficiency. This value was chosen by measuring the number of images that could be processed per second. The Figure 4.1a shows that using an NVIDIA GeForce GTX 1080 Ti graphics card for training, the efficiency increased from the batch size of 1 to 32 but decreased when $B > 32$. For this reason, the value $B = 32$ was chosen as a fixed value throughout the training.

Adam let us define the learning rate, which decides how much the weights can be moved in the opposite direction of the gradient for a specific batch. Choosing a value for the learning rate is a trade-off process between training time and training



(a) The number of images that is processed per second during the training phase, with different batch sizes (B).



(b) The cost function with a specific learning rate, using ReLU as activation function.

performance. If the learning rate is too low, the optimization will take a lot of time due to the small steps towards the minimum of the loss function. If the learning rate is set to a high value, it could cause undesirable divergent behavior and put constraints on training to converge.

Smith [38] proposed a technique for strategically selecting a learning rate for a neural network. By periodically incrementing the learning rate, starting from a low value, and plot the corresponding loss value for each value of η. Before exploring the learning rate η, a fixed batch size B = 32 was used, as these parameters are interdependent. A graph after training the network one epoch was created, using ReLUs as activation and a learning rate in the scale of $10^{-7} < \eta < 10^{-1}$. In Figure 4.1b, the two points with the fastest decrease in loss but remained stable was chosen as the learning rate. The results, as shown in Figure 4.1b, indicate that the point for the specific batch size B = 32, lies between the values 10^{-6} and 10^{-5}.

4.1.2 Regularization

Throughout the project, different techniques were used to regularize the models. One technique applied during training was early stopping. In Keras, one can set a number of parameters to terminate the training process when the model stop to improve. The loss was monitored with a patience of 5 epochs, which means that if the loss had not decreased in 5 epochs, the training was aborted. This was used with the intention to prevent an overfitted model. Another efficient method to ensure that the best model is saved during the training phase, is to apply Keras "ModelCheckpoint", where the parameter save_best_only set to true saves a model as soon as it has improved. In this case, the validation loss of the model was monitored as the deciding factor.

A widely used technique to improve the model is to apply dropout. As mentioned in Paragraph 2.1.7, dropout help to prevent overfitting the model and a common

value to use for dropout is 0.5, which means that every neuron has a 50% chance to be dropped(skipped). A dropout value of 0.5 has proven to be a successful value, hence we chose this as a static value throughout the project.

4.2 Training

During the project, the problem with a relatively small dataset led to a constant fight against the overfitting problem. As mentioned, several methods were practiced to reduce overfitting, such as early stopping, regularization techniques and data augmentation. Two measurements monitored during the training phase were the cost function evaluated on both the validation and training data, and the validation accuracy compared to the training accuracy. These together can provide valuable information about how the training process is elapsing.

The Figure 4.2 shows 30 epochs of training VGG16 without any regularization and any data augmentation. Figure 4.2 shows a clear trend of an overfitted model after epoch eight. Closer inspection of the table shows that the training loss (blue line) constantly continues towards a lower value until it reaches close to zero. Different from the training loss, the validation loss(orange line) starts to increase at the point of epoch 10, which is a typical indication of an overfitted model.

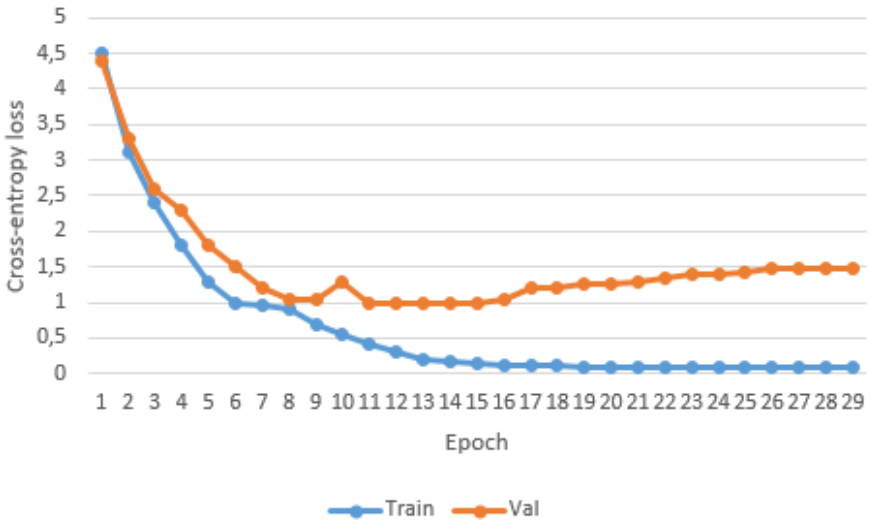


Figure 4.2: The figure shows a typical graph of an overfitted model. The training loss is the blue line and validation loss is in orange, both displayed as a function of the number of epochs. When the validation loss increases(the orange line) while the training loss decreases(blue line), then a situation of overfitting is observed.

4.3 Model performance

In order to compare the models between each other, and to other approaches, they have to be evaluated on the separate testset. After setting the hyperparameters, the

networks were modified and trained with two separate dataset. One dataset con et. After extracting the most successful model for each approach, they were evaluated on the testset. The classification statistics are presented in the Table 4.1.

Network architecture	Parameters	TPR	TPR*	FPR	FPR*	A	A*	Speed
VGG16	15,242,050	0,95	0,98	0	0,02	0,97	0,97	56fps
InceptionV3	23,903,010	0,95	0,97	0,03	0,03	0,96	0,95	59fps
DenseNet	20,291,138	0,94	0,95	0	0,09	0,97	0,93	26fps
Firenet	92,857,730	0,91	0,93	0,06	0,04	0,92	0,94	89fps

Table 4.1: The table shows the difference between the architectures implemented in the thesis and the difference in accuracy when training with augmented data and the original data. (* = Model trained on augmented data).

To evaluate the different models, the true positive rate (TPR), false positive rate (FPR), accuracy (A) and the speed(frames processed per second) were analyzed, for the reasons stated in Section 3.4. The Table 4.1 presents the comparisons of the networks using augmented data (*) or non-augmented data. It is apparent from this table that the TPR for VGG16, DenseNet and Firenet had a small increase when they were trained on the augmented data. However, they also had a small increase in terms of FPR. The DenseNet is the newest and deepest network of the presented ones and achieved good scores in terms of detection. What is interesting about the data in this table is that VGG16 had very high performance in this binary classification task.

The experiment was conducted on an Intel Core i7-8700 3.20 GHz CPU and 32 GB of RAM. The resulting frame processed per second is displayed in Table 4.1 together with the parameter complexity. The VGG16, inceptionV3 and DenseNet have roughly the same parameter complexity, while the Firenet is the network that stands out with its 90M parameters. There was a significant difference in terms of speed as the Firenet could process roughly 90 frames per second, while the VGG16 and InceptionV3 could process 1/3 less. The DenseNet could process 26 frames per second with the complexity of 20M parameters. The natural explanation for this is that the speed, as well as the performance of a CNN, does not only depend on the complexity, also the depth.

4.3.1 Multimodal architecture vs single stream model

A multimodal architecture can be constructed in different ways. They can be merged in an early, middle or late stage, which is described in Section 2.2. During this experiment, several merge techniques was tested, namely "add", "concatenate" and average. The performance of this experiment is shown in Table 4.2.

Merge algorithm	TPR	FPR	Acc	Speed
Add	1	0	1	28fps
Concatenate	1	0	1	28fps
Average	1	0.03	0.98	29fps

Table 4.2: The table shows the performance between the merge algorithms that were applied when fusing the VGG16 network(RGB) and Firenet(thermal).

As presented in Table 4.1 and Table 4.2 the multimodal architecture achieved better results than all the presented single streamed networks, except in the category speed. Since the presented multimodal approach utilize inputs from two modalities, better coverage than if only one modality was used is expected. However, it should be mentioned that the results from VGG16 are very similar to the multimodal approach. Gaining roughly an increase 25 fps from using only RGB sensors to losing accuracy of the model is not a valid defense, especially when adding a thermal sensor input to the system make the system detect fire even though it is dark, which is not covered in the single streamed testset. Looking at Table 4.2 it is clear that adding or concatenating the RGB model and thermal model is the most sufficient use of merging a multimodal when training a CNN to detect fire.

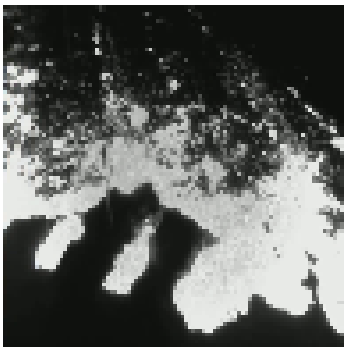
The RGB and thermal model are trained on images with the quadratic size of 200x200. When testing the 200x200 dimensions as they were for the separate networks, the model speed decreased a lot. This is due to the natural cause of handling multiple inputs, which required double amount of computational power.

As described, the multimodal network requires multiple inputs, one RGB and one thermal image. To precisely evaluate this model, the testset needs to be constructed in a way where the RGB image matches the corresponding thermal image from the exact same scene. An example of a "fire" input to the network is shown in Figure 4.3. The amount of videos where an RGB and a thermal camera is displaying the same scene in real-time is limited, which resulted in a relatively small testset. The experiment conducted was done using a testset compromised of 300 images for the multimodal approach.

4.3.2 Classifying images

Some of the original baseline methods used for this project, presented a set of images that were misclassified. To measure the presented approach versus other original baseline methods, misclassified images taken from baseline methods were extracted and evaluated on our model. Zhao et al. [50], with the approach of combining a

(a) RGB image from the scene at the time T.



(b) Thermal image from the same scene, at time T

Figure 4.3: A "fire" class input for the multimodal network.

(a) Example of an RGB image that were misclassified by the single streamed CNN (VGG16).

(b) Correctly classified if the multimodal network is fed with additional thermal information.

Figure 4.4: Misclassified by the RGB architecture (a). Correctly classified by the multimodal architecture, by feeding the network with the additional information of the thermal image, (a) and (b).

saliency detection and deep learning based strategy presented an issue with mist present in images. They describe the equality between mist and smoke features and present a set of misclassified images Figure 4.5, mostly of negative samples. The images were correctly classified by the single RGB stream (InceptionV3) network. The result from the classification for each images, is presented in the Figure 4.5. The multimodal approach was not tested on these images, due to the fact that the corresponding thermal images from the same scene does not exist.

(a) Our approach classified this image as 83% "notFire".

(b) Our approach classified this image as 79% "notFire".

(c) Our approach classified this image as 98% "notFire".

Figure 4.5: Images that were misclassified by the method proposed by Zhao et al. [50], and correctly classified by our method (RGB stream based on InceptionV3).

4.4 Suitable for a drone application

In recent years, Nvidia has developed and improved their Jetson series, where the TX2 and AGX Xavier are the two most recent ones. According to Nvidia Developer Blog, AGX Xavier deliver 20 times the computing power compared to the TX2 and up to 28 fps per watt compared to the TX2 which only read 2 fps per watt when classifying images on VGG-19. Moreover, microchip computers such as Raspberry Pi are even less powerful.

Computer	CPU	RAM	GPU
Raspberry Pi 3b	Quad Core 1.2GHz	1 GB	-
Jetson TX2	Dual-Core Nvidia Denver 2	8GB	256-core NVIDIA Pascal
Jetson AGX Xavier	8-core ARM v8.2	16GB	512-core Volta
Thesis Computer	Intel i7-8700	32 GB	Gigabyte 1080 TI

Table 4.3: Comparison of computers that could potentially be used in a fire detection system.

CNN's spend most of their computational time on the convolutional layers and most of the storage is allocated to the fully connected layers. To accelerate the process of matrix multiplication, GPU's are optimal. Devices similar to the Raspberry Pi, with 2GB of RAM, are essentially too weak to deploy deep CNN models. Our algorithm is operating in real-time but does not require high fps. This is due to the fact that the drone will move on a high altitude and with a limited speed. However, the Raspberry Pi is too slow and does not meet the memory requirements, even for this task.

The Jetson TX2 is among the high-end microchip computers and designed to allow developers to run deep learning on mobile in real-time. Since the models used in this thesis are large, the need for high computational power and a large RAM memory is required. To run the fire detection system on a Jetson TX2 should be possible

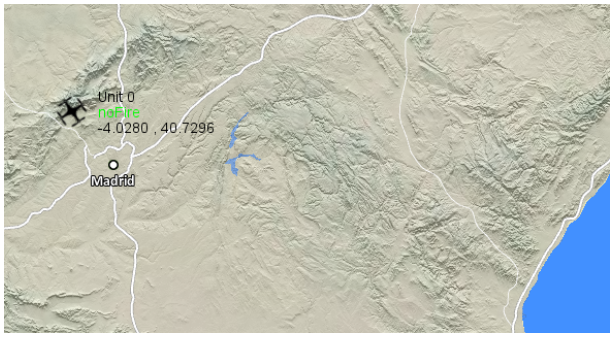
due to its large RAM memory and high computation power.

The Jetson AGX Xavier is the successor of the Jetson TX2 and has a tremendous performance upgrade, both in terms of computational power and memory. Xavier is given the ideal prerequisites for deploying computer vision systems. The Xavier can process roughly 200 images per second using VGG-19, with an input tensor of 224x224 and a batch size = 1.

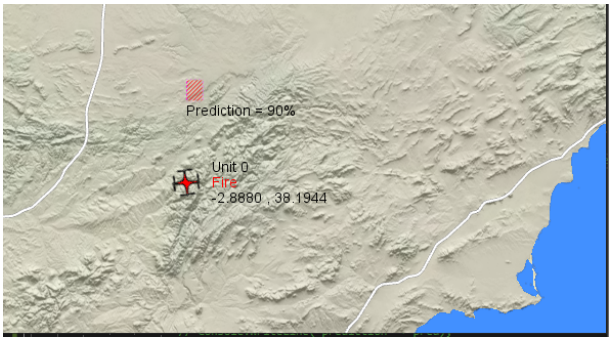
As shown in Table 4.2 on page 35, when running the multimodal fire detection on the thesis computer listed in Table 4.3 on the previous page 56 fps for VGG16 was achieved, which is more than enough to perform fire detection. The predecessor to Nvidia Jetson TX2 was compared to an Intel i7 6700 skylake, which had roughly equal performance in terms of image processing in deep learning models [23].

4.5 Visualization

As shown in Figure 4.6a, each drone initialized on the map is running the fire detection algorithm and is updated every 25 ms. The drones read every frame and classify them as either "fire" or "not fire". When a frame is classified as "fire" the fire detection application transfer the frame, coordinate information and detection probability to a database, shown in appendix A.3.



(a) A UAV hovering an area in Spain, without detecting any fire. The GPS position of the drone is shown, as well as the status of the drone.



(b) The same UAV detected a fire, which is indicated by changing the state of the UAV and the creation of a fire polygon feature is initialized.

Figure 4.6: An example of how a drone and its findings are visualized in the Fire Visualizer. The detected fire is shown as a interactable fire polygon together with the prediction probability and the drone in the different states depending on if a fire is found or not.

The fire visualizer provides an overview of the drones findings, making it easy to locate the detected fire. The application is delivered with a simple GUI for interacting with the finding of the drones, which follow most of the standard interaction rules for maps.

5
Discussion

In this chapter a discussion about the thesis result will be conducted. The reasoning behind applying a multimodal architecture and its advantages are discussed. Furthermore, the use of a fire detection system implemented on a UAV is reviewed.

5.1 Fire detection using CNN

In the result chapter, it was documented that a multimodal architecture outperform a single stream CNN in detection rates while losing approximately 25 fps. In many cases, the speed of the network is critical to a real-time application. However, as in the case when it is not, the prediction rates of the model are more interesting. The performance found in Table 4.2 on page 35 compared to Table 4.1 on page 34 should be enough of an argument that applying a multimodal architecture is more efficient than using only a single stream CNN. The multimodal coverage will make the system more stable and maintain stable predictions regardless of the time of the day and weather. However, because of the lack of multimodal data, the testset consisting of multimodal images have been limited and to conclude any definitive results, a wider set of test data would be needed. The data acquired in this theses are mostly from day time, which could potentially limit the model from classifying fire when dark. Evaluation of the multimodal architecture has not been performed when it is dark. A beginning of a wildfire with only smoke features, could lead to the images captured by the RGB camera to be completely black. However, if a fire would escalate at night, an RGB camera would most likely detect the fire since it creates light.

Even though the multimodal architecture has achieved promising results, using one drone to locate the area of a fire is not sufficient. The use of several drones would map the area of a fire faster as well as cover a larger area which could result in a system of higher quality. However, a long-range thermal camera is expensive and mounting them on each drone could result in a cost that is higher than what the system would generate in return. A reasonable approach to tackle this issue could be to wait until long-range thermal cameras have become cheaper or to implement the system using only RGB cameras, even though a multimodal approach has shown to be more efficient.

5.1.1 Misclassified images from original baseline methods

Figure 4.5 on page 37 show how the original baseline method from Zhao et al. [50] misclassify the images of mist as "fire" where our method classify them as "notFire". The reason they gave for the misclassified images, is that "smoke and mist are highly similar in both color and shape. Even the human eye can mistake these images for smoke due to the similar features". A reason behind the results could be that images of clouds, mist and fire images in our trainset have been included. The fire images in the trainset include both features of fire and smoke. However, the exact reason why our method classify the images as the corresponding true labels and Zhao et al. misclassify the images is impossible to determine.

5.1.2 Comparison between single stream and multimodal

As shown in Section 4.3.1 on page 35, the multimodal approach outperform the single stream approach. In Figure 4.4 on page 36 the single stream approach predict the image as "notFire" while the multimodal approach predict the image as "fire". The RGB image probably mistake this image for a cloud, due to the very vague feature combined with no presence of actual fire. The multimodal gets fed with both the RGB and thermal image, making it more clear that this is a fire, since the thermal image clearly shows that there is heat in the image. The RGB image is still vaguely classified as "notFire" and the thermal image is strongly classified as fire. Hence, the images is classified as a fire.

5.2 Performance on mobile

Most of the leading network architectures utilize a large number of layers which make it problematic to deploy this method on a budget microchip computer such as a Raspberry Pi. However, most of the processing power that is needed when working with CNNs are used while training the networks. Therefore, to use a pre-trained model in an application require much less computing power.

The need for powerful microchip computers has long been a priority for researchers in order to achieve deep learning tasks on mobile. With microchip computers such as Jetson TX2 and AGX Xavier it is possible run the system presented in this thesis. The GPU on mentioned devices can load the models and predict if the image contain fire and send the predicted "fire images" to the fire visualizer. An advantage of using a powerful microchip computer on a drone to predict each image, is that sending each image to be predicted by a server could be expensive and the transfer algorithm might not be able to send all the images that are recorded by the drone in real time.

An alternative solution would be to simply use the drones as information collectors. One could let the drone act as a client, sending information such as the frame and metadata to a server, where the actual image processing is done. A server-client solution could be possible with additional hardware for sending the video stream to

the computer. However, since the UAV's driving system has to be on-board which is probably computationally expensive, the most natural solution is that the video frame computations is on-board as well.

A major advantage of having the computation on-board is that the fire detection system gets more robust. A server-client model typically make the system heavily dependent on a stable connection between the server and the client. If the computation is made on-board, the detected fire frames could easily be saved into the memory and transfer that information when the connection is stable. However, storing large amount of frames and forward them all when the connection is stable, could cause delays in the system and memory allocation problems.

6
Conclusion

This chapter will relate back to the research questions and draw conclusions based on the discoveries in the result and discussion sections. Moreover, suggestions for future work are presented.

6.1 Research questions

To answer the stated research questions, an experimental setup was designed, consisting of two separate datasets and several network architectures that have been evaluated. During the experiment, several network architectures were modified, trained and deployed. In the first stage of the experiment, two separate modalities were tested, namely RGB inputs and thermal inputs, and in the second with a multimodal approach for classifying fire. Moreover, a comparison of our approach and the baseline methods is evaluated. The experiments will be concluded by reflecting back to the initial questions, which are:

Could a two-streamed multimodal CNN's binary classifier outperform previous approaches that are explicitly based on color values and temporal information in a fire in terms of detection rates?

With the single streamed CNN based on RGB images, an accuracy of 97% with a 2% false positive rate was recorded. The two-streamed multimodal network is a slightly modified VGG16 (RGB) that is fused with a modified FireNet (thermal), running the fire detection algorithm on 28 fps with a TPR of 100% and a FPR of 0%. The previous approaches have stated an accuracy rate between 83 and 98. However, due to the fact that there exist no reference dataset with the right quality nor size, the previously presented approaches as well as ours, remains vague.

Could data augmentation on the relatively small dataset increase the performance of the classification model.

The original dataset consists of 20 000 RGB images. The images passed through the pre-processing step, making them compatible as input data for the CNN. The augmented dataset consists of 70 000 RGB images, where the general augmentation techniques such as rotation, shifting, zoom, flip and fill was deployed on the original dataset.

The result of the augmentation led to more stable training curves and better fitted models. Surprisingly, there was no great performance increase by using the augmented dataset. However, most of the architectures had a small increase in both accuracy and TPR when they were trained on the augmented dataset. This may be due to the high performance of the original dataset, that consisted of roughly 10 000 image samples per class.

How does a two-streamed multimodal CNN perform compared to using a single streamed CNN?

As expected, the two streamed CNN outperformed the single streamed CNN in terms of detection. Furthermore, the multimodal network requires to process two images in order to make a classification, naturally this resulted in a lower FPS compared to the single steam network. The reason why the multimodal network makes such accurate classification is due to the fact that thermal cameras are designed to find heat changes. However, one has to balance the trade off between speed, accuracy and the price of choosing a long-range thermal camera in the multimodal approach.

Is this approach appropriate for the hardware equipped on a UAV?

Due to the need of powerful microchip computer that have been a priority for researchers the last decade and the interest of convolutional neural networks on mobile have increased, the performance of these microchip computers has increased significantly. These are designed to handle different computer vision techniques, such as a convolutional neural network. Moreover, the fire detection algorithm runs in real-time, however, there it should be no need for an fps higher than 10, which make our system compatible with some of the presented microchip computers.

6.2 Future work

This approach could be further analyzed by configure different networks and try to make them more optimized towards fire detection. Adding more advanced techniques could potentially improve the performance of the single streamed CNN. In this thesis, we experimented with a multimodal approach, where we applied a late fusion model. However, a comparison between early, late and hybrid fusion was not conducted. Therefore, further research needs to examine more closely the links between early, hybrid and late fusion.

In limitations, it was stated that the thesis will not consider any work with the drone-hardware. During the thesis, it was assumed that the drone is equipped with high-end hardware and is semi-autonomous. There exist many proposed algorithms for semi-autonomous drones, however not so optimized for forest monitoring. Here, the question of energy consumption arises and is of great importance. This would certainly be a fruitful area for further work as the need for forest fire monitoring is constantly increasing and here is, therefore, a definite need for an algorithm handling the drone route efficiently.

During the thesis, data from real wildfire scenarios was provided by Valencian Agency for Security and Emergency Response, which made this project possible. However, the solution is not tested on real drone hardware in forests. Hence, a natural progression of this work is to analyze this approach on a drone equipped with the suggested hardware in 4.4.

Bibliography

[1] Hamed H. Aghdam and Jahani H. Elnaz. *Guide to Convolutional Neural Networks - A Practical Application to Traffic-Sign Detection and Classification*. SPRINGER INTERNATIONAL PU, 2017, p. 282. ISBN: 9783319861906.

[2] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis Philippe Morency. *Multimodal Machine Learning: A Survey and Taxonomy*. 2019. DOI: 10.1109/TPAMI.2018.2798607. URL: https://arxiv.org/pdf/1705.09406.pdf.

[3] Patrick Behrer. "Why We Need Forests". In: *Sense and Sustainability* (2012). URL: http://www.senseandsustainability.net/2012/02/05/why-we-need-forests/.

[4] Jason Brownlee. *How to Choose Loss Functions When Training Deep Learning Neural Networks*. 2019. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[5] SR Cajal and Nobel Lecture. "The structure and connexions of neurons". In: *Nobel Lectures* (1967). URL: http://courses.biology.utah.edu/bastiani/3230/DB%20Lecture/Handouts/Lec%2014%20Neuro/cajal-lecture.pdf.

[6] Turgay Celik et al. "Fire detection using statistical color model in video sequences". In: *Journal of Visual Communication and Image Representation* 18.2 (Apr. 2007), pp. 176–185. ISSN: 10473203. DOI: 10.1016/j.jvcir.2006.12.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1047320306000927.

[7] Che-Bin Liu and Narendra Ahuja. "Vision based fire detection". In: *International Conference on Pattern Recognition*. 2004, pp. 134–137. DOI: 10.1109/icpr.2004.1333722. URL: https://experts.illinois.edu/en/publications/vision-based-fire-detection.

[8] *Classification Of Neurons By Structure And Function*. 2017. URL: http://physiologyplus.com/classification-of-neurons-by-structure-and-function/.

[9] Daphne Cornelisse. *A Comprehensive Guide to Convolutional Neural Networks*. 2018. URL: https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050.

[10] Henry Cruz et al. “Efficient forest fire detection index for application in Unmanned Aerial Systems (UASs)”. In: *Sensors (Switzerland)* 16.6 (June 2016), p. 893. ISSN: 14248220. DOI: 10.3390/s16060893. URL: http://www.mdpi.com/1424-8220/16/6/893.

[11] Andrew J Dunnings and Toby P Breckon. “Experimentally defined convolutional neural network architecture variants for non-temporal real-time fire detection”. In: *Proceedings - International Conference on Image Processing, ICIP*. 2018, pp. 1358–1362. ISBN: 9781479970612. DOI: 10.1109/ICIP.2018.8451657. URL: https://breckon.org/toby/publications/papers/dunnings18fire.pdf.

[12] Chi Thang Duong, Remi Lebret, and Karl Aberer. “Multimodal Classification for Analysing Social Media”. In: (2017). URL: https://arxiv.org/pdf/1708.02099.pdf.

[13] Andreas Eitel et al. “Multimodal deep learning for robust RGB-D object recognition”. In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2015-Decem. 2015, pp. 681–687. ISBN: 9781479999941. DOI: 10.1109/IROS.2015.7353446. URL: https://arxiv.org/pdf/1507.06821.pdf.

[14] GISGeography. *World Geodetic System (WGS84) - GIS Geography*. 2018. URL: https://gisgeography.com/wgs84-world-geodetic-system/.

[15] Hamza Mahmood. *Softmax Function, Simplified – Towards Data Science*. 2018. URL: https://towardsdatascience.com/softmax-function-simplified-714068bf8156.

[16] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015, pp. 1026–1034. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.123. URL: http://ieeexplore.ieee.org/document/7410480/.

[17] Katie Hoover. *Wildfire Statistics*. Tech. rep. 2018, p. 2. URL: https://fas.org/sgp/crs/misc/IF10244.pdf.

[18] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, pp. 2261–2269. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.243. URL: http://ieeexplore.ieee.org/document/8099726/.

[19] Walter Phillips Iii, Mubarak Shah, and Vitoria Lobo. “Flame recognition in video q”. In: 23 (2002), pp. 319–327. URL: http://crcv.ucf.edu/projects/Fire/Fire.pdf.

[20] Mark Z. Jacobson. “Effects of biomass burning on climate, accounting for heat and moisture fluxes, black and brown carbon, and cloud absorption effects”. In: *Journal of Geophysical Research: Atmospheres* 119.14 (July 2014), pp. 8980–9002. ISSN: 2169897X. DOI: 10.1002/2014JD021861. URL: http://doi.wiley.com/10.1002/2014JD021861.

[21] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: (June 2014). URL: http://arxiv.org/abs/1408.5093.

[22] Keras. *Keras Documentation*. 2017. URL: https://keras.io/.

[23] Lee Mathews. *Nvidia says its new Jetson board can compete with a Core i7*. 2015. URL: https://www.geek.com/chips/nvidia-says-its-new-jetson-board-can-compete-with-a-core-i7-1639215/.

[24] Ihab S. Mohamed et al. "Detection, localisation and tracking of pallets using machine learning techniques and 2D range data". In: (Mar. 2018). URL: http://arxiv.org/abs/1803.11254.

[25] Muneeb ul Hassan. *VGG16 - Convolutional Network for Classification and Detection*. 2018. URL: https://neurohive.io/en/popular-networks/vgg16/.

[26] A. NAMOZOV and Y. I. CHO. "An Efficient Deep Learning Algorithm for Fire and Smoke Detection with Limited Data". In: *Advances in Electrical and Computer Engineering* 18.4 (2018), pp. 121–128. ISSN: 1582-7445. DOI: 10.4316/AECE.2018.04015. URL: http://www.aece.ro/abstractplus.php?year=2018&number=4&article=15.

[27] Pablo Ruiz Ruiz. *Understanding and visualizing ResNets – Towards Data Science*. 2018. URL: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a.

[28] Kumarguru Poobalan and Siau-Chuin Liew. "Fire Detection Algorithm Using Image". In: December (2015), pp. 12–13. URL: http://worldconferences.net.

[29] Public Safety. *Carmenta*. 2019. URL: https://www.carmenta.com/en/industries/public-safety.

[30] PyTorch. *PyTorch*. URL: https://pytorch.org/.

[31] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. "Large-scale deep unsupervised learning using graphics processors". In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. New York, New York, USA: ACM Press, 2009, pp. 1–8. ISBN: 9781605585161. DOI: 10.1145/1553374.1553486. URL: http://portal.acm.org/citation.cfm?doid=1553374.1553486.

[32] J Ramiro Martínez-De Dios, Luis Merino, and Aníbal Ollero. *FIRE DETECTION USING AUTONOMOUS AERIAL VEHICLES WITH INFRARED AND VISUAL CAMERAS*. Tech. rep. 2005, pp. 660–665. DOI: 10.3182/20050703-6-CZ-1902.01380. URL: https://ac.els-cdn.com/S147466701637392X/1-s2.0-S147466701637392X-main.pdf?_tid=7d8416c4-9326-486d-810b-fa14917a1cac&acdnat=1549006970_fb1293d7aa3393e98e3a62f22ceec723.

[33] Eyal Reingold. "What are artificial neural networks?" In: (). URL: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html.

[34] Shaeke Salman and Xiuwen Liu. "Overfitting Mechanism and Avoidance in Deep Neural Networks". In: (2019). URL: https://www.youtube.com/watch?v=Qi1Yry33TQE.%20http://arxiv.org/abs/1901.06566.

[35] Shubham Jain. *An Overview of Regularization Techniques in Deep Learning (with Python code)*. 2018. URL: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/.

[36] Sik-Ho Tsang. *Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015*. 2018. URL: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c.

[37] Karen Simonyan and Andrew Zisserman. *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. Tech. rep. 2015. URL: http://www.robots.ox.ac.uk/.

[38] Leslie N. Smith. "Cyclical learning rates for training neural networks". In: *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*. June 2017, pp. 464–472. ISBN: 9781509048229. DOI: 10.1109/WACV.2017.58. URL: http://arxiv.org/abs/1506.01186.

[39] Nitish Srivastava et al. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Tech. rep. 2014, pp. 1929–1958. URL: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[40] Ilya Sutskever et al. *On the importance of initialization and momentum in deep learning*. Tech. rep. 2013. URL: https://www.cs.toronto.edu/~fritz/absps/momentum.pdf.

[41] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-Decem. 2016, pp. 2818–2826. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.308. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf.

[42] TechnoReview. *Artificial Neural Network : Beginning of the AI revolution*. 2018. URL: https://hackernoon.com/artificial-neural-network-a843ff870338.

[43] TensorFlow. *TensorFlow*. URL: https://www.tensorflow.org/about.

[44] The University of Queensland. *What is a neuron? - Queensland Brain Institute - University of Queensland*. 2018. URL: https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron.

[45] Vibhor Nigam. *Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning*. URL: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90.

[46] Jörg Wagner et al. "Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks". In: *Conference: 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* April (2016), pp. 27–29. URL: https://www.ais.uni-bonn.de/papers/ESANN_2016_Wagner.pdf.

[47] Chi Yuan et al. “Aerial Images-Based Forest Fire Detection for Firefighting Using Optical Remote Sensing Techniques and Unmanned Aerial Vehicles”. In: *J Intell Robot Syst* 88 (2017), pp. 635–654. DOI: 10.1007/s10846-016-0464-7. URL: https://link-springer-com.proxy.lib.chalmers.se/content/pdf/10.1007%2Fs10846-016-0464-7.pdf.

[48] Matthew D. Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8689 LNCS. PART 1. 2014, pp. 818–833. ISBN: 9783319105895. DOI: 10.1007/978-3-319-10590-1{_}53. URL: http://link.springer.com/10.1007/978-3-319-10590-1_53.

[49] Qi-xing Zhang et al. “Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images”. In: *Procedia Engineering* 211 (2018), pp. 441–446. ISSN: 18777058. DOI: 10.1016/j.proeng.2017.12.034. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877705817362574.

[50] Yi Zhao et al. “Saliency Detection and Deep Learning-Based Wildfire Identification in UAV Imagery”. In: *Sensors* 18.3 (Feb. 2018), p. 712. ISSN: 1424-8220. DOI: 10.3390/s18030712. URL: http://www.mdpi.com/1424-8220/18/3/712.

A

Appendix

A.1 Dataset structure

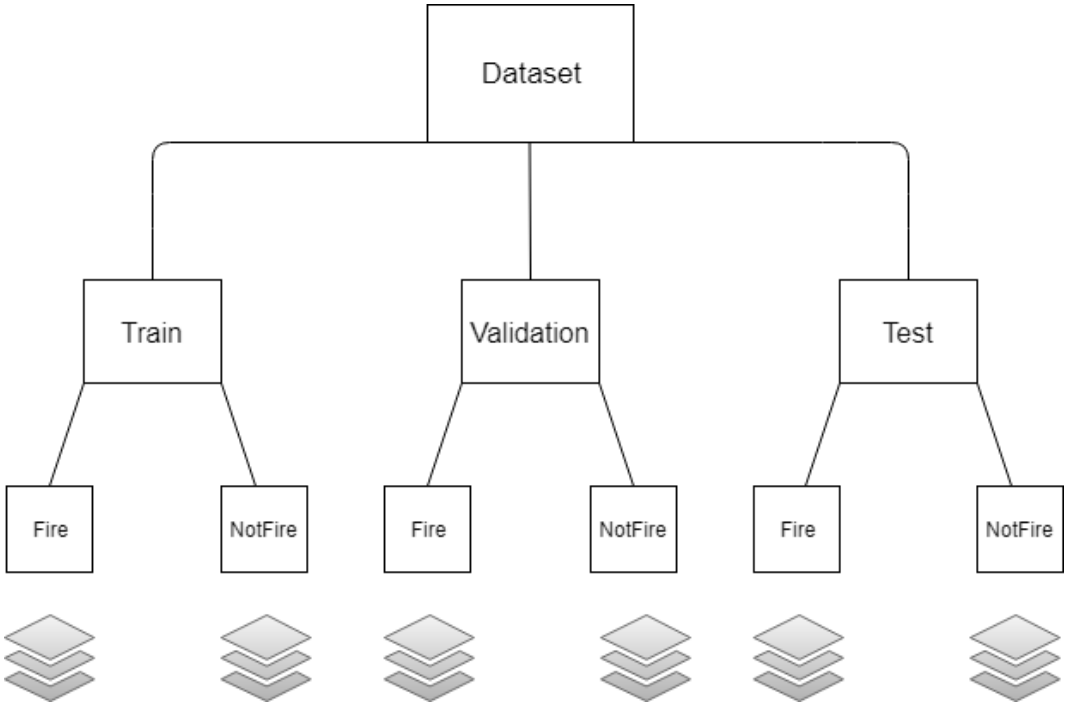
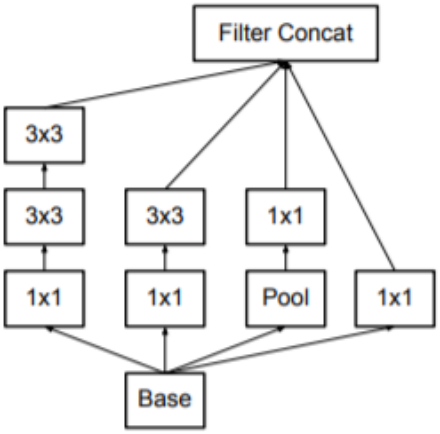


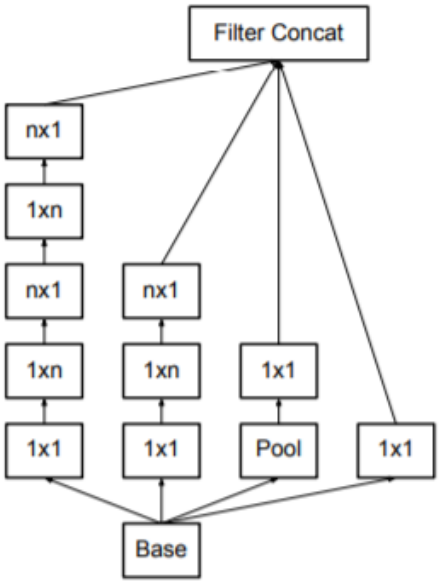
Figure A.1: How the train, validation and testdata are structured in Keras.

A.2 Inception modules

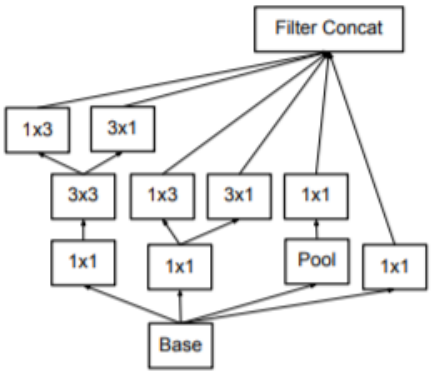
A visualization of the inception modules A, B and C.



(a) Inception module A. By replacing a 5x5 filter with two 3x3 filters the number of parameters are reduced by 28%.



(b) Inception module B. Each 3x3 filters are replaced with 2 nx1 and 1xn filters.



(c) Inception module C. By replacing a 3x3 filter to 3x1 and 1x3 filters the number of parameters are reduced by 33%.

Figure A.2: The different Inception Modules [41].

A.3 System structure

Figure A.3 show the structure of the system presented in the thesis.

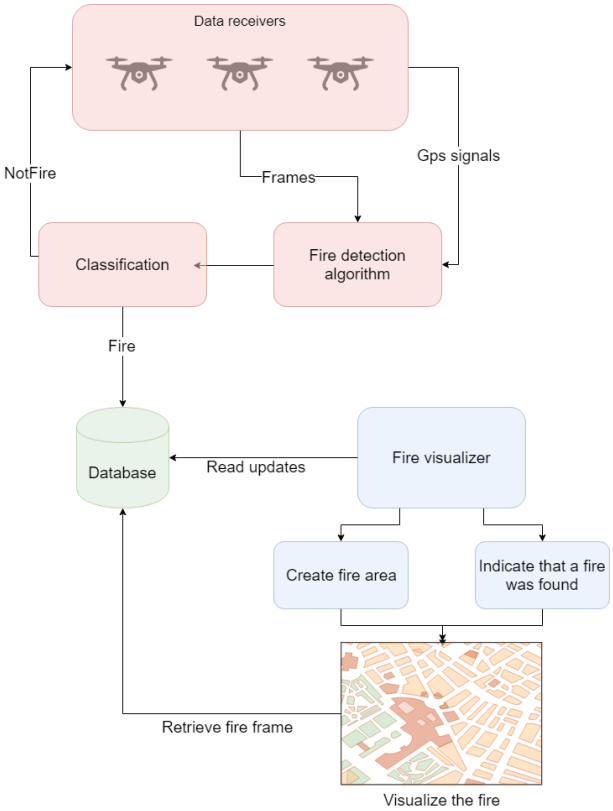


Figure A.3: The visualization application's relation to rest of the architecture.