Fire and Smoke Detection using Fully Supervised Training Methods and Search by QuadTree

Gonçalo de Carvalho Rodrigues Callet Perrolas

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Ricardo Adriano Ribeiro
Prof. Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Ricardo Adriano Ribeiro
Member of the Committee: Prof. José Manuel Peixoto do Nascimento

February 2021
Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
Abstract

Wildfires are one of the most destructive and uncontrollable natural disasters faced by man-kind. The fire-fighting combat teams can greatly benefit from a reliable source of information about the different aspects within a fire scene. By locating, in real time, the current position of the fire fronts, a better fighting strategy can be developed to help the efficiency of the operations. It is also critical, the detection of the early fire ignitions to take action as soon as possible.

This work proposes an original deep learning method for fire and smoke detection using aerial images of wildfires. We trained both a full image classifier and a segmentation network in order to detect the presence of fire/smoke and to localize the regions of the images containing said phenomenon. Along side the detection component, we used an innovative Quad-Tree algorithm to increase the precision of the detections, processing the images into smaller patches.

The proposed system was able to produce segmentations with a high level of precision and detail, achieving an Avg. IoU of 0.88 for fire and 0.83 for the smoke class on a test aerial image dataset. The system proved to be highly capable of being used on a real fire scenario.

Keywords

Fire Detection; Smoke Detection; Aerial Images; Wildfire; Convolutional Neural Networks.
Resumo

Os incêndios são uma das catástrofes naturais mais destrutivas e incontroláveis enfrentadas pela humanidade. As equipas de combate a incêndios podem beneficiar imensamente de uma fonte de informação fiável sobre os diferentes aspectos relacionados com um incêndio. Ao localizar, em tempo real, a posição actual das frentes de incêndio, uma melhor estratégia de combate pode ser desenvolvida de modo a ajudar eficiência das operações. É também fundamental, a detecção das primeiras ignições de incêndio de modo a tomar medidas o mais rapidamente possível.

Este trabalho propõe um método original de aprendizagem profunda para a detecção de fogo e fumo recorrendo a imagens aéreas de fogos florestais. Treinámos tanto um classificador de imagem como uma rede de segmentação a fim de detectar a presença de fogo/fumo e de localizar as regiões das imagens que contêm os respetivos fenómenos. Associada à componente de detecção, utilizámos um algoritmo inovador de Quad-Tree para aumentar a precisão das detecções, processando as imagens em parcelas mais pequenas.

O sistema proposto foi capaz de produzir segmentações com um elevado nível de precisão e detalhe, conseguindo um Avg. IoU de 0,88 para o fogo e 0,83 para a classe de fumo num conjunto de imagens aéreas de teste. O sistema provou ser altamente capaz de ser utilizado num cenário de incêndio real.

Palavras Chave

Deteção de Chama; Deteção de Fumo; Imagens Aéreas; Incêndios Florestais; Redes Neurais Convolucionais.
# Contents

1 Introduction ........................................... 2
   1.1 Motivation .......................................... 4
   1.2 Objectives .......................................... 4
   1.3 Outline of the Document........................... 5

2 Related Work and State-Of-The-Art ....................... 6
   2.1 Related Work ........................................ 7
      2.1.1 Classic Methods ................................ 7
      2.1.2 Deep Learning Methods ......................... 8

3 Neural Networks .................................... 10
   3.1 Neural Networks .................................... 11
   3.2 Training Neural Networks ......................... 12
      3.2.1 Loss Functions ................................ 13
      3.2.2 Optimization Algorithms ....................... 14
   3.3 Convolutional Neural Networks ..................... 16
      3.3.1 Feature Learning ............................... 17
      3.3.2 Classification ................................ 19
   3.4 Semantic Segmentation CNNs ....................... 19

4 Methodology ...................................... 21
   4.1 Overview ......................................... 22
   4.2 Search by QuadTree ................................ 23
   4.3 Classification Network (SqueezeNet) ............... 25
   4.4 Segmentation Network (U-Net) ....................... 28

5 Experimental Setup ................................ 30
   5.1 Datasets ......................................... 31
   5.2 Network Training .................................. 33
      5.2.1 SqueezeNet (Classification) ................. 33
      5.2.2 U-Net (Segmentation) ......................... 35
5.3 Metrics .......................................................... 37
5.4 Tools Used ...................................................... 38
5.5 Experiments Catalog ........................................... 38

6 Results .......................................................... 40
   6.1 Experiment A.1. .............................................. 41
   6.2 Experiment A.2. .............................................. 42
   6.3 Experiment B .................................................. 43
   6.4 Experiment C .................................................. 46
   6.5 Experiment D .................................................. 49

7 Conclusions ..................................................... 53

A Code of Project .................................................. 59
List of Figures

1.1 Firefront Operations Diagram .................................................. 3
3.1 Structure of a neural network and its neurons ................................. 11
3.2 Visualization of the optimization problem, \( w_1 \) and \( w_2 \) are weights belonging to a layer. . 14
3.3 Structure of a Convolutional Neural Networks (CNN). ...................... 16
3.4 Characteristics of the Convolutional Layer, the kernel. ..................... 17
3.5 Types of pooling ................................................................. 18
3.6 Structure of the U-Net ........................................................... 20
4.1 Overall structure of the proposed system. ..................................... 22
4.2 Large Fire Area has 23.7% of ratio ......................................... 24
4.3 Small Fire Area has 1.73% of ratio ........................................ 24
4.4 SqueezeNet Architecture and corresponding Fire Module .................. 25
4.5 SqueezeNet fire output examples .............................................. 26
4.6 SqueezeNet smoke output examples .......................................... 27
4.7 Structure of the U-Net ........................................................... 28
4.8 U-Net output examples .......................................................... 29
5.1 Examples of images from the dataset. ........................................ 31
5.2 Training progress, evaluating the SqueezeNet models accuracy and loss .......................... 34
5.3 Training progress, evaluating the U-Net models accuracy and loss ........ 36
6.1 Study on classification performance with different thresholds. ............ 41
6.2 Study on segmentation performance with different thresholds. ............ 42
6.3 Example in which patch processing is not beneficial ........................ 44
6.4 Example in which patch processing is beneficial ............................ 45
6.5 Box plot graphic of Avg. IoU distribution on fire segmentation ............ 47
6.6 Box plot graphic of Avg. IoU distribution on smoke segmentation ........ 48
6.7 Fire tests on Firefront Dataset images ................................. 49
6.8 Tests on other images taken from a long distance .................... 50
6.9 Smoke test on a Firefront Dataset image ............................... 51
6.10 Test on Image taken from a long distance .............................. 51
6.11 Tests on other Smoke Images ........................................... 52
List of Tables

3.1 Commonly used Neuron Activation Functions ............................................ 12
3.2 Loss Functions ......................................................................................... 13
5.1 Overview of the dataset ............................................................................. 32
5.2 Training Performance Results ................................................................. 34
5.3 Training Performance Results ................................................................... 36
5.4 Example of a Confusion Matrix for two class detection ............................. 37
5.5 Experiments Catalog .................................................................................. 39
6.1 Study of the patch size in the segmentation performance for fire and smoke ................................................. 43
6.2 Overall Performance Comparison of the complete system vs. system with missing components, on fire segmentation ................................................................. 46
6.3 Overall performance comparison of the complete system vs. system with missing components, on smoke segmentation ........................................................................ 48

List of Algorithms

1 Quad_Tree Algorithm ..................................................................................... 23
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Convolutional Neural Networks</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicles</td>
</tr>
<tr>
<td>ADAI</td>
<td>Fire Propagation Models</td>
</tr>
<tr>
<td>ANEPC</td>
<td>National Coordination Center</td>
</tr>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>R-CNN</td>
<td>Region-based Convolutional Neural Networks</td>
</tr>
<tr>
<td>TP</td>
<td>True Positives</td>
</tr>
<tr>
<td>TN</td>
<td>True Negatives</td>
</tr>
<tr>
<td>FP</td>
<td>False Positives</td>
</tr>
<tr>
<td>FN</td>
<td>False Negatives</td>
</tr>
<tr>
<td>IoU</td>
<td>Intersection over union</td>
</tr>
<tr>
<td>SD</td>
<td>Standard Deviation</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Motivation ................................................. 4
1.2 Objectives .............................................. 4
1.3 Outline of the Document ................................. 5
This work is related to the Firefront project \(^1\). This project has the objective of developing a system intended to help the firefighting teams in forest fires providing them with information obtained through the automatic detection and tracing of the fire fronts in real-time. The detection is done using aerial images taken from unmanned aerial vehicles or fire fighting vehicles equipped with cameras that will fly over the fire scenes. In figure 1.1 is exposed a diagram of the all the operations within the Firefront project.

The images taken by the vehicles will then be transmitted into ground stations where the detection process will occur. The consequent detection results will act as base information on the remaining operation components. The Fire Propagation Models (ADAI) can be applied and the strategies developed by the National Coordination Center (ANEPC) can be further reinforced in order to take better decisions to help the fire fighting teams on site.

On this work, we will focus on the detection of the two most important phenomenons occurring within a wildfire, fire and smoke. The fire detection is important while trying to localize the current fire fronts on the terrain, this detection is related more directly to the task of real time monitoring of the fire in parallel of the fire fighting operations. The smoke detection, on the other hand, is essential on the early phases of the wildfire, in which the aerial vehicles can operate with a recon/patrol strategy while trying to detect the early fire ignitions.

\(^1\)http://www.firefront.pt/
1.1 Motivation

Forest fires represent one of the biggest catastrophes affecting Portugal and many other countries. Every year, a large number of hectares are burnt due to the difficulties that the firefighting teams face when trying to deal with the fire. Some times, the brave work of the combat teams is not able to be effective enough in order to counter the propagation of the fire, which will eventually result in big proportion wildfires.

Every agent involved in the fire combat scene make use of all the information within their reach to take the best decisions they can, being the current position of all the fire fronts one of the most important. After geo-referencing all the fire fronts, predictive methods of fire propagation can be used in order to allocate more efficiently the combat resources (both humans and material). This also contributes to the safety of the people that are willing to take risks to fight the fire.

At the early stages of fire ignitions, smoke can be very important in order to detect the occurrence of fire as early as possible. Large columns of smoke are produced by the initial ignitions and they can be seen from the sky, making drones also capable of detecting them.

Every year we see a tremendous evolution in areas related with deep learning. A lot of different day-to-day tasks are using some type of neural network in order to solve some issue otherwise difficult to solve. Applying this types of technology together with intelligent aerial vehicles such as drones we can make the process of locating the fire fronts more efficient and automatic.

1.2 Objectives

On this thesis we will address all the processing that is going to be done to the images captured by the drones that fly above the forest fires.

Firstly the images are taken from drones equipped with Red Green Blue (RGB) cameras and a transmission system able to send those images. Then, image processing techniques involving deep neural networks are used in order to detect fire and smoke on those images. It is important to choose an adequate network for the proposed objective, leading to the best results possible to make the system viable and capable of being used with real life fire images taken from drones. The final results we want to come up with, using our proposed methods, consist of the semantic segmentation of the fire and smoke classes within the images taken.
1.3 Outline of the Document

This thesis is is organized as follows:

• Chapter 1 covers a brief introductory discussion about this thesis topic, where the motivations and the main objectives to be met are described;

• On Chapter 2 is presented the current state-of-the-art of the methods and tools used and the different approaches used to solve tasks similar to the ones addressed in this thesis work;

• Chapter 3 addresses a small overview of neural networks, describing them and its components and how they will be useful to this work;

• Chapter 4 exposes the methodology used to achieve the proposed objectives doing initially a brief overall description of all the components of the system and how they relate to each other;

• On Chapter 5 are exposed the setup methods that will enable the realization of tests to our system;

• All the testing experiments done while developing this work are represented in Chapter 6 together with all the results produced by the system components that indicate the true advantages of the methods used;

• Finally, on Chapter 7, the conclusions are stated together with the thesis achievements and possible future improvements to this work.
Related Work and State-Of-The-Art
Over the last two decades, automatic methods of forest fire fighting technologies have shown an increasing popularity as a research topic. The different approaches and methods used in detection problem can be divided into two different groups: methods that make use of classic image processing techniques where images are analyzed regarding textures, contrasts or RGB components and the second group involves all the techniques that resort to neural networks to do the fire or smoke detection.

2.1 Related Work

Several efforts have been made in the development on the topic of using Unmanned Aerial Vehicles (UAV) with vision-based systems to monitor, detect, and fight forest fires [1], using classic methods of image processing or deep learning to extract the relevant features that define a fire or a smoke area.

From the research done, we also noticed that none of the works reviewed mention the multi-scale problem that occurs while using neural networks to detect objects with a wide size variance. This is an interesting topic that will be discussed more in depth throughout this work. This will also act as a motivation pushing towards the creation of a method that tries to solve this challenge.

2.1.1 Classic Methods

The vast majority of classic methods are based on algorithms that make use of the three RGB components of each pixel because the color histogram for fire regions is distinguishable from the background scenario. However, the same cannot be said about the smoke due to high similarities with other objects like clouds.

Different indexes can be calculated according to the RGB values of the pixels in order to enhance the areas of fire to then apply a threshold to binarize the images into the respective classes [2]. There are a lot of different algorithms intended for fire segmentation using different color spaces [3] however, by doing an overview benchmark of their performance [4], we can conclude that they tend to be very biased in regard of the camera characteristics, the environment conditions and other external factors. The performance also shown average results that highly fluctuate depending on the datasets used for testing.

We find that the large majority of fire/smoke detection system that are operating rely on image captured by fixed cameras on high altitude in order to cover large areas of forest terrain. One of the most used techniques in this types of systems is Background Subtraction, where the elements that do not belong to the background are segmented from the images. One example of this types of systems operating in Portugal is the project CICLOPE [5], that monitors 1,300,000 hectares of forest and detects the occurrence of fire doing smoke analysis on the images captured. Similarly, smoke detection can be done through its color, its motion and time domain changes. [6]. This type of techniques can be applied whenever the cameras are fixed, which is not our case.

In form of conclusion, classical methods tend to be very efficient, where the processing time required
is relatively low. However, the problem with hand-crafted features is that it requires a higher level of expertise and a lot of fine-tuning for specific problems. Changing the environment and the cameras parameters will induce a lot of false detections on the system.

Very recently, the appearance of deep learning has completely revolutionized many areas related with image processing and remote sensing, by using neural network more specifically, Convolutional Neural Networks (CNN) [7].

2.1.2 Deep Learning Methods

This approach has many advantages, the main one being that it allows the collection of characteristics of an image in a more abstract and complex way, being able to evaluate which are the characteristics that define a flame or smoke. This strategy requires a mandatory stage, the network is trained to make it suitable for the detection task in question. It is also important to mention that there are different types of training, depending on the label level of the dataset used. Labels can have a pixel wise identification for each different class (fully supervised) or a more general image level label where is stated if the image contains a specific type of class (weakly supervised).

As stated earlier, the most commonly used network for this types of tasks is CNN. We can find examples of the use of CNN in fire and smoke detection by weakly supervised training [8]. The results show that the resulting network created was over-fitted to the data used in training. This is an issue caused by one of the most relevant obstacles in this research topic, the lack of datasets containing fire and smoke images with the respective labelling.

Other strategies can be used in collaboration with CNNs. One example is trying to use the moving dynamic of fire in time domain to extract more features from the fire [9]. The author compared the results using the AlexNet [10], VGG16 [11] and SqueezeNet [12], and they all showed to be able to correctly detect fire. Using the temporal information did not improve the performance significantly and the heat-maps produced while trying to localize the fire regions seem to be very vague and not well defined. The classification networks were trained with a large dataset of image level labels (160 000 imgs.) and they achieved a high level of detection (classification) performance.

There are several CNN extensions, one of which is Region-based Convolutional Neural Networks (R-CNN) [13] which can be used in the detection task as it brings advantages over network efficiency on searching for fire or smoke instances [14]. This technique may only detect one instance of fire/smoke per image, making it not suitable for detecting multiple inter-lapping areas on the image. The dataset used by this author was really interesting and it shown potential to be used in this thesis work. More information about the Corsican Fire Dataset [15] will be given on Chapter 5.

These machine learning techniques are able to surpass the limitations of the traditional methods, described earlier. Therefore we consider the use of deep learning to be appropriate for the detection task.
in hand.
3

Neural Networks

Contents

3.1 Neural Networks ......................................................... 11
3.2 Training Neural Networks ............................................. 12
3.3 Convolutional Neural Networks ..................................... 16
3.4 Semantic Segmentation CNNs ...................................... 19
Since the basis of deep learning comes from a system called neural networks, we will briefly introduce it, in the following section, and then move on to more advanced and appropriate methods for multi-class semantic segmentation.

### 3.1 Neural Networks

A Neural Network is a computational system inspired by the biological function and structure of a human brain. Similarly to our brains these systems consist of groups of several connected neurons. The function of each artificial neuron belonging to a neural network can be mathematically modelled as:

\[
y = f(\phi(x, w))
\]

\[
\phi(x, w) = w_0 + \sum_{i=1}^{p} w_i x_i
\]

having input values of \( x = (x_1, x_2, ..., x_i) \) and each element \( x_i \) is fed and then multiplied by a respective adjusted weight \( w_i \). After combining all the inputs together with a \( w_0 \) bias, an activation function \( f \) is applied, resulting on the output \( y \).

Groups of neurons are organized into layers connected to each other, forming the neural network, like is shown in Figure 3.1.

![Structure of a neural network and its neurons](image)

**Figure 3.1:** Structure of a neural network and its neurons
As stated earlier, each neuron has a activation function at the end of its model. This function can be linear or non-linear and it allows the neural network to learn to model more complex functions. The most commonly used activation functions are shown in Table 3.1.

<table>
<thead>
<tr>
<th>Activation Function</th>
<th>Formulation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid</td>
<td>$f(x) = \frac{1}{1 + e^{-x}}$ (3.2)</td>
<td>Output values bound between 0 and 1, normalizing the output of each neuron</td>
</tr>
<tr>
<td>Softmax</td>
<td>$f_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}$ (3.3)</td>
<td>Able to handle multiple classes. The output for each class ranges between 0 and 1, giving the probability of the input value being in a specific class. Useful for output neurons, typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories;</td>
</tr>
<tr>
<td>Hyperbolic Tangent</td>
<td>$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (3.4)</td>
<td>The output ranges from -1 to 1. This function is usually used on hidden layers, and it models easier more diverse and mean zero data by having a zero centered output;</td>
</tr>
<tr>
<td>ReLU (Rectified Linear Unit)</td>
<td>$f(x) = \begin{cases} x &amp; \text{if } x \geq 0, \ 0 &amp; \text{if } x &lt; 0 \end{cases}$ (3.5)</td>
<td>Constant gradient for positive inputs. ReLU has a derivative function and allows for backpropagation. Enables the network to learn quicker, but when inputs approach zero, or are negative, the gradient of the function becomes zero and makes learning harder.</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>$f(x) = \begin{cases} x &amp; \text{if } x \geq 0, \ \alpha x &amp; \text{if } x &lt; 0 \end{cases}$ (3.6)</td>
<td>Similar to ReLU by adding a small positive linear slope for the negative values. Has the benefit of enabling the learning for negative input values also, allowing backpropagation. ($\alpha &lt; 1$)</td>
</tr>
</tbody>
</table>

### 3.2 Training Neural Networks

Neural Networks are able to approximate accurately any complex function, given a big set of layers containing neurons. Throughout the training phase, the neuron weights $w_i$ suffer adjustments making the network suitable for the task in hand. The small adjustments to the weights are done using backpropagation.
3.2.1 Loss Functions

In order to assess the final scores produced by the network, a loss function is used to evaluate the error between the produced predictions and the expected results (labels). Given a set of inputs \( x = (x_1, x_2, ..., x_i) \), expected outputs (labels) \( y = (y_1, y_2, ..., y_i) \), and predictions made by the network \( \bar{y} = (\bar{y}_1, \bar{y}_2, ..., \bar{y}_i) \), the goal of the training stage is to minimize the error between \( y \) and \( \bar{y} \).

There are several loss functions that can be used, depending on the type of problem. Table 3.2 shows the most relevant loss functions used.

**Table 3.2: Loss Functions**

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Formulation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Entropy</td>
<td>( L = -\sum_{c=1}^{C} \bar{y}_c \log(y_c) ) (3.7)</td>
<td>( c ) represents each class, and ( C ) the total number of classes. Cross-Entropy is widely used for classification purposes, like pixel level classification in segmentation. There are two types: binary cross-entropy if there are only two classes and categorical cross-entropy if ( C &gt; 2 );</td>
</tr>
<tr>
<td>Weighted Cross-Entropy</td>
<td>( L = -\sum_{c=1}^{C} w_c \bar{y}_c \log(y_c) ) (3.8)</td>
<td>This loss function is a cross-entropy variant. In this variant, a weight ( w_c ) is added. It is widely used in case of unbalanced data between classes, by choosing to favor one class over the others;</td>
</tr>
<tr>
<td>Focal Loss</td>
<td>( p_t = \begin{cases} p &amp; \text{if } c = 1, \ 1 - p &amp; \text{if otherwise} \end{cases} ) (3.9) ( L(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) ) (3.10)</td>
<td>Focal Loss proposes to down-weight easy examples and focus training on hard negatives using a modulating factor, ( (1 - p_t)^\gamma ). It works well for highly imbalanced class scenarios. ( (\gamma &gt; 1) );</td>
</tr>
<tr>
<td>Dice Loss</td>
<td>( d_c = 1 - \frac{2 \sum_{\text{pixels}} \bar{y}<em>p y_p}{\sum</em>{\text{pixels}} y_p^2 \sum_{\text{pixels}} \bar{y}_p^2} ) (3.11)</td>
<td>Each class is normalized separately using the dice coefficient ( (d_c) ), to help the training using unbalanced data.</td>
</tr>
</tbody>
</table>

\( L = \sum_{c}^{C} d_c \frac{1}{C} \) (3.12)
3.2.2 Optimization Algorithms

The training stage of a neural network can be seen as an optimization problem where we try to minimize the loss function by updating the weights $w_i$ iteratively. This optimization problem can be illustrated as shown in Figure 3.2. Next, we will briefly discussed some of the most relevant optimization algorithms used for training neural networks.

Figure 3.2: Visualization of the optimization problem, $w_1$ and $w_2$ are weights belonging to a layer.

- **Gradient Descent**

  This algorithm updates the weights with

  $$w_{i+1} = w_i - \eta \frac{\partial L}{\partial w}$$  \hspace{1cm} (3.13)

  where $\eta$ is the learning rate constant. The partial derivatives $\frac{\partial L}{\partial w}$ are calculated using the backpropagation method. Each helps to find which are the weights that are contributing more significantly to the error or loss $L$.

- **Stochastic Gradient Descent (SGD)**

  The conventional gradient descent methods takes a long time to converge in larger datasets, as they have a very high computational cost.

  By using Stochastic Gradient Descent, only a random subset of samples from the training data are chosen in each run to update parameter during optimisation, instead of using every single element. This method converges to the intended solution more quickly.
• **SGD with Momentum**

Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a factor $\gamma$ to the update vector $\Delta w$ of the past time step to the current update vector:

$$\Delta w_t = \gamma \Delta w_{t-1} + \eta \frac{\partial L}{\partial w}$$

(3.14)

$$w_{i+1} = w_i - \Delta w_t$$

(3.15)

The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions.

• **Adagrad**

This algorithm adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features. For this reason, it is well-suited for dealing with sparse data. The weights are updated with

$$w_{t+1,i} = w_{t,i} - \eta \frac{\partial L_{t,i}}{\sqrt{G_t} + \epsilon}$$

(3.16)

$G_t \in \mathbb{R}^{d \times d}$ here is a diagonal matrix where each diagonal element is the sum of the squares of the partial derivatives $\frac{\partial L_{t,i}}{\partial w_{t,i}}$ up to time step $t$, while $\epsilon$ is a smoothing term that avoids division by zero (usually on the order of $1e^{-8}$). One of Adagrad’s main benefits is that it eliminates the need to manually tune the learning rate.

• **AdaDelta**

Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size.

• **Adam**

Adam is one of the most used optimization algorithms used in deep learning. It has a similar logic to SGD with momentum, where we compute the decaying averages of past and past squared gradients $m_t$ and and $v_t$ respectively as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w}$$

(3.17)
\[ v_t = \beta_2 v_{t-1} + (1 - \beta_2) \frac{\partial L^2}{\partial w} \]  
(3.18)

(where \( \beta_1 \) and \( \beta_2 \) are close to 1) and they are used to compute the bias-corrected first and second moment estimates:

\[ \hat{m} = \frac{m_{i+1}}{1 - \beta_1} \]  
(3.19)

\[ \hat{v} = \frac{v_{i+1}}{1 - \beta_2} \]  
(3.20)

to be able to update the weight parameters with

\[ w_{i+1} = w_i - \eta \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}} \]  
(3.21)

using again a small coefficient \( \epsilon \) to avoid divisions by zero.

### 3.3 Convolutional Neural Networks

Within the area of neural networks, the CNN plays a strong role in solving problems related to image processing. The scientific advances that have occurred in the association between deep learning and image processing have been built on CNN architectures. This type of neural networks receives an image as input. The image is then processed, where the relevant characteristics are extracted through different types of filters (layers), each with a particular function. Finally, the product of this process will feed a structure capable of producing the final classification decision. In Figure 3.3, a scheme of the most common structure of a CNN is presented.

![Figure 3.3: Structure of a CNN. (from the web)](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 27 December 2019, 19:58)
We can divide the CNN algorithm into two distinct parts, one that computes features on the image and the final one that does classification. Each of these has specific filters and functions that will be addressed below.

3.3.1 Feature Learning

On this stage a set of layers or filters are used:

**Convolutional Layers - The Kernel**

The convolutional layer is the first filter that extracts information from the image. Convolution is a process that maintains the relationship between pixels when learning by using small portions (small squares) of the input data. This process is based on a sequential mathematical operation between a portion of the image and a matrix filter (feature map) capable of activating certain characteristics of the image. For an image segment with the dimensions \((H \times W \times D)\), where \(D\) represents the different channels of the image (RGB), for a filter \((H_f \times W_f \times D_f)\) the multiplicative process results in a matrix \((H - H_f + 1) \times (W - W_f + 1) \times 1\), since the filter matrix (the Kernel) sweeps the whole image, this movement is expressed in Figure 3.4(a).

![Figure 3.4: Characteristics of the Convolutional Layer, the kernel.](image1)

(a) Kernel Movement. In red we can see the filter running through all dimensions of the image

(b) Kernel Multiplicative Process. The image is represented with green, in yellow the Kernel filter and the resulting matrix with pink

The multiplication that is performed covers the elements of the kernel filter matrix and the elements of a certain segment of the image and then, the sum of all the elements of the resulting matrix is made. This result corresponds to one of the elements of the final matrix. The multiplicative process is exemplified in the figure 3.4(b), where the green numbers represent the image, the yellow represents the kernel filter (with the values of its matrix in red) and, in pink, we have the matrix resulting from sweeping the whole image.
**Activation Function - Relu (non-linearity)**

The rectifying layer, using the Relu activation function, is used to introduce non-linearity to help the neural network learn more quickly using more complex and large datasets. The Relu will also help to solve the vanishing gradient problem when data has values close to zero.

**Pooling Layers**

Like the convolution process, the pooling filter is responsible for reducing the size of the matrices, reducing the computational power required for the processing. However, even using this sub-sampling technique, it is possible to keep the important information that was extracted along all the filters previously applied.

The pooling processes can be of several types, being the most common:

- **Maximum Pooling**, where is extracted the maximum value of the segment;

- **Average Pooling**, where is calculated the mean of the elements of the respective patch;

- **Sum Pooling**, where is done the sum of all the elements of that portion of the matrix;

These pooling types are represented on Figure 3.5.

![Figure 3.5: Types of pooling](image)

The main objective of the convolutional operation is to extract high-level features from the image. As the number of filter layers in the network increases (convolution, non-linearity and pooling sets) the extracted features become more and more abstract. The first filters are responsible for collecting the low-Level characteristics of the images, such as outlines, colors or gradient orientations. Each of the Kernel filters implemented in a neural network has the role of activating certain features that will be fundamental for the final classification tasks.
3.3.2 Classification

After a series of convolution and pooling operations the resulting matrix (feature map) is transformed into a vector. It is now intended, in this final phase, to make the classification among the desired classes of the images (or segments of them) through the previously extracted high-level features.

**Fully connected layer**

To reach this we will use fully connected layer. This type of filter is very similar to convolution filters in that they also have a multiplicative functional structure. The main difference is that the neurons of the convolutional filters are only connected to a local region of the input and the neurons of the fully connected layer are connected to all the activations of the previous filter. The purpose of this type of filter is to analyze the feature map and determine which features are more correlated with a certain type of class.

The common CNN, that uses fully connected layers at its end, produces, as output, a vector with the size of the number of different classes we want to train the network in. These tools enable the classification on image level. If a class appears on a certain image then the network will come up with a high probability for that class.

In this thesis work we want to do the localization of fire/smoke portions on images, so using classic CNN models is not enough for the end objective. There is a special type of CNN responsible for doing classification on pixel level, or semantic segmentation.

3.4 Semantic Segmentation CNNs

Semantic segmentation refers to the process of linking each pixel in an image to a class label. Normal CNNs include a lot of pooling and convolutional layers. The feature maps of the network tend to decrease in size consecutively from one layer to the next but for semantic segmentation this is not ideal. It is necessary for the output feature maps to have the same size as the input image so it is essential to recover the details lost by the down-scaling process. For solving this issue there is the need to create a method to up-scale the feature maps previously determined by the network to the original size of the input image.

One of the most used network in semantic segmentation problems is U-Net \[16\]. This network was originally created to be used in biomedical image processing while trying to surpass the challenge of scarce number of datasets containing images for training (this will also be a challenge to this thesis work). As illustrated in Figure 3.6 \[16\], the network architecture is made up of a contracting path on the left and an expansive path on the right. The contracting path is made up of two $3 \times 3$ convolutions. Each of the convolutions is followed by a rectified linear unit layer and a $2 \times 2$ max pooling operation for down-
sampling. Every down-sampling stage doubles the number of feature channels. The expansive path steps include an up-sampling of the feature channels. This is followed by $2 \times 2$ up-convolution that halves the number of feature channels. The final layer is a $1 \times 1$ convolution that is used to map the component feature vectors to the required number of classes. Which gives it the so-called u-shaped architecture.

![Figure 3.6: Structure of the U-Net](image)

The expansive path is able to up-scale the information gathered by contracting part and provides an extra spatial reference to the features detected. As an end result, this allows for the pixel wise segmentation of the input image into the different chosen classes.
4 Methodology

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Overview</td>
<td>22</td>
</tr>
<tr>
<td>4.2 Search by QuadTree</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Classification Network (SqueezeNet)</td>
<td>25</td>
</tr>
<tr>
<td>4.4 Segmentation Network (U-Net)</td>
<td>28</td>
</tr>
</tbody>
</table>
In this section we will expose the methods used in order to achieve the proposed objectives. Initially, a brief description of the overall system will be done followed by a more complete characterization of each different system’s component.

### 4.1 Overview

As stated on previous chapters, the proposed system is intended to process images taken from an aerial point of view, either taken from conventional aerial vehicle like an airplane or helicopter or an unmanned vehicle like a drone. Those vehicles must be equipped with a RGB camera with enough resolution to be able to detect small fire/smoke areas and a transmission system to send the images to a processing unit on land. The aerial pictures will then go through our system in order to detect the areas of fire and smoke within them.

The overall structure of all the system components is represented in figure 4.1. The first section of the system is the QuadTree block that dynamically analyse the image, diving it into patches depending on the results of the final segmentation stage. The main purpose of this algorithm is to solve the multi-scale problem of the detection and, depending on the size of the fire/smoke area on the image, the algorithm will tend to either do a more precise detection, involving smaller size patches, or a more global search using the all image. The QuadTree block will output the complete input image or a portion of it.

![Figure 4.1: Overall structure of the proposed system.](image)

The output of the QuadTree algorithm will then move along to the detection stage that includes the sequence of a classification network followed by a segmentation network. It is important to state that the detection system for fire and smoke are independent, this means there will be two different pairs of classification and segmentation networks, each one more suitable for the respective class to be detected.

The portions of the images are initially fed as input to the classification network (SqueezeNet) in order to detect the presence of the phenomenon (either fire or smoke) and, if the output of the classification is positive, that image patch goes through to the segmentation network (U-Net) in order to detect the
regions of the image containing said class, represented by a binary image.

The binary result will then be used in the QuadTree to evaluate if it is necessary to do a more precise detection on the current analysed patch. If this is not necessary then the process is ended and the final result is reached.

### 4.2 Search by QuadTree

The QuadTree terminology [17] refers to a method of dividing certain data structures in which each element has exactly four children. We intend to apply the same logic to the search for the patch of the image that contains fire or smoke. The algorithm will have the global image as the starting point and then, if needed, will recursively use smaller and smaller patches, searching for the respective phenomena. In each iterative step, the current analysed patch will be divided into four smaller segments in order to do a more precise detection.

All the logic of the system’s flow is controlled within this functional block and his algorithm behavior is explained in Algorithm 1.

**Algorithm 1: QuadTree Algorithm**

```plaintext
Function QuadTree(Input_Patch):
    positive = Classify(input_patch);
    if positive then
        input_patch_seg = Segment(input_patch);
        if Ratio(input_patch_seg) < max_ratio AND Size(input_patch) > min_size then
            patches = Split_in_4(input_patch);
            for patch in patches do
                result_i = QuadTree(patch_i)
                GluePatches(results, final_result);
        else
            if positive then
                final_result = input_patch_seg;
            else
                final_result = blank_patch;
    return final_result
```

Inside the algorithm there are calls to the same function (recurrence) to be able to process every patch desired and to produce an end result as an aggregate of said patches. The function has an RGB image as an input and will output the gray-scale result of the detection iterations.
There are some relevant parameters that control the behavior of the QuadTree algorithm:

- **Ratio**: Represents the ratio between the number of pixels belonging to positive and negative cases. If a large area of fire is segmented, then the value of that ratio will be bigger than for a smaller fire. The Max_Ratio parameter is used to control how big a fire or smoke must be, in a certain patch, to stop the "zooming" process. In Figure 4.2 we can see an example in which there is no need to do a more precise segmentation due to big size of the fire. After applying the segmentation on a global image level the value of the ratio was 23.7%. On the other end of the spectre, a small area of fire, as shown in Figure 4.3, will produce smaller ratios (in this case 1.73%) which can mean that the algorithm should do a more precise detection, using smaller patches.

![Figure 4.2: Large Fire Area has 23.7% of ratio](image)

- **Min_Size**: Expresses the minimum size of patch the algorithm can reach. This parameter can be helpful to adjust the system to the camera equipped on the vehicle. If the camera has a big resolution sensor, then even smaller patches can have a lot of information.

![Figure 4.3: Small Fire Area has 1.73% of ratio](image)
4.3 Classification Network (SqueezeNet)

The patches produced by the QuadTree will then move on to the detection stage of the system, starting with a classification stage. This stage includes a classification network called SqueezeNet [12]. With equivalent accuracy as Alex-Net [10], this network has efficiency advantages due to its smaller CNN architecture. For example, they are more feasible to deploy on FPGAs and other hardware with limited memory and require less bandwidth to export a new model from the cloud.

In general, this network has fewer parameters to be optimized on the training stage which makes it more suitable for situations in which the dataset used is considerably small, like in our case, while trying to maintain high accuracy values. In figure 4.4 [18] it is represented the typical structure of the SqueezeNet as well as its Fire Module.

![SqueezeNet Architecture](image1)

(a) SqueezeNet Architecture

![Fire Module](image2)

(b) Fire Module

**Figure 4.4:** SqueezeNet Architecture and corresponding Fire Module

To achieve the desired efficient performance there are some fundamental architectural design strategies to take into account that define this network:

- The large majority of the network convolutional filters are 1x1, since a 1×1 filter has 9× fewer parameters than a 3×3 filter for example;
- The use of a Fire Module. A Fire module is comprised of a squeeze convolution layer (which has only 1×1 filters), feeding into an expand layer that has a mix of 1×1 and 3×3 convolution filters. We can then decrease the number of input channels to 3×3 filters using the respective squeeze layers. This module has some similarities with the Inception Module present in GoogleNet [19];
- Finally, down sampling late in the network so that convolution layers have large activation maps that can lead to higher classification accuracy.
The main purpose of using a classification network before doing segmentation on each patch is to add a filter that can help to reduce the number of false positives. The network will output a probability array with the size corresponding to the number of classes. As stated before, fire and smoke detection tasks are independent, which means there are going to be two different instances of this network, one to detect the presence of fire and another for smoke.

In case of fire, the classification network is going to help filtering out negative cases like rooftops, sunsets and other reddish shades that can appear is certain patches. In Figure 4.5 we can see six different outputs from the SqueezeNet, represented as a probabilistic array containing the probability for each class (\([\text{Fire Prob.}, \text{Negative Prob.}]\)).

![Figure 4.5: SqueezeNet fire output examples. Firstly the class predicted, next the probabilistic output array with \([P(\text{Fire}), P(\text{Negative})]\) and finally the image used.](image)

The same applies for the smoke detection task. In this case, the hard negative cases for the network consist mainly of clouds, as they are hardly distinguishable from a smoke, even for a human sometimes.
Some examples of outputs of the network for smoke and non-smoke images are shown in Figure 4.6. The probabilistic array contains \([ \text{Negative Prob.}, \text{Smoke Prob.}]\).

**Figure 4.6:** SqueezeNet fire output examples. Firstly the class predicted, next the probabilistic output array with \([P(\text{Negative}), P(\text{Smoke})]\) and finally the image used.

After the training stage, it is necessary to determine the best threshold to binarize the classification probabilities. This study is shown in more detail on next chapters.
4.4 Segmentation Network (U-Net)

The next step, in our detection flow, consists of the segmentation stage. Every patch that resulted on a positive detection by the classification network goes through the U-Net, a segmentation network that was already introduced in Chapter 3. The purpose of this network is to localize the areas corresponding to each class on a pixel level, the output consists of a gray-scale image in which each pixel contains the probability of belonging to a certain class.

Similarly as the classification network, we will have two different instances of this network, one to be associated with the SqueezeNet dedicated to fire and other for the one dedicated for smoke. The structure of the U-Net is shown in Figure 4.7.

![Figure 4.7: Structure of the U-Net](image)

The network comprises of two different paths, the contraction/encoder path and the expansion/decoder path. The contraction path consists of a repeated application of a 3x3 convolutions (unpadded) each followed by a ReLU and a 2x2 max pooling operation with stride 2 for down-sampling. This can help to extract more advanced features but it also reduce the size of feature maps.

The expansion path consists of the up-sampling of the feature map produced in the contraction path. It consists of consecutive 2x2 convolution (“up-convolution”) that halves the number of feature channels and a 3x3 convolutions followed by a ReLU. After each up-conv., we also have concatenation of feature maps from the contraction with the same level. This helps transferring feature localization information from the contraction path to the expansion path.
In figure 4.8, are shown some examples of u-net outputs, both for the fire class and smoke class.

![Figure 4.8: U-Net output examples](image)

The output of the segmentation network consists of a gray-scale image that represents the areas where the corresponding class is well defined. Each image pixel will range in value between zero and one (normalized), representing the probability of the pixel containing that class.

Similarly as the SqueezeNet, it is necessary to evaluate the detection threshold to limit the detection areas, this study is done in Chapter 6.

After applying the designated threshold, the binary image produced will then be feed into the QuadTree logic block to move on to the next iteration step.
5

Experimental Setup

Contents

5.1 Datasets .............................................................. 31
5.2 Network Training .................................................. 33
5.3 Metrics ............................................................... 37
5.4 Tools Used .......................................................... 38
5.5 Experiments Catalog ............................................... 38
In this chapter we will discuss setup procedures taken to optimize each system component and the different tests that were taken in consideration to evaluate the performance of the overall system.

5.1 Datasets

In order to train the chosen networks and to do subsequently some performance test there is a need to gather a set of images containing the adequate labelling for the respective network.

For the classification networks the labelling needed is on image level, where each image has to be identified if it contains the phenomenon or not. Regarding the datasets needed for the segmentation networks, the labels have to specify each pixel class. In total, four different types of datasets were gathered.

The images gathered for the fire segmentation dataset mainly came from three different sources: Corsican Dataset [15] (RGB images with pixel wise labelling), smaller datasets found online and a batch of images gathered online that were manually labelled to extend as much as possible the size of the dataset. The smoke segmentation dataset consists also of datasets with pixel wise labels found online [20] [21] and some more images segmented manually. In figure 5.1 we show examples of four different images paired with the respective labelling for fire and smoke cases.

![Figure 5.1: Examples of images from the dataset.](image-url)
The nomenclature negative identifies that the image does not include the respective phenomenon. The hardest negatives for the network to distinguish in case of fire are images containing sunsets/sunrises, reddish color sky, red foliage and red objects that can appear in fire fighting situations (like fire trucks, airplanes and helicopters). For smoke, the main negative object that is extremely hard to distinguish are clouds. So, in order to properly train the network against these cases it is important to include in the datasets images containing these difficult negative cases.

The datasets for classification include the ones used for the segmentation training together with some additional images. In Table 5.1 we present a small overview of the datasets used for the training phase where is identified the corresponding number of images.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Fire Positive</th>
<th>Fire Negative</th>
<th>Smoke Positive</th>
<th>Smoke Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>800 imgs</td>
<td>520 imgs</td>
<td>500 imgs</td>
<td>300 imgs</td>
</tr>
<tr>
<td>Segmentation</td>
<td>Containing Fire 700 imgs</td>
<td>Containing Smoke 300 imgs</td>
<td>450 imgs</td>
<td>60 imgs</td>
</tr>
</tbody>
</table>

The number of images gathered in the datasets goes to show how hard it was to find complete and adequate datasets containing pixel segmentation. Doing manual segmentation is a very demanding task but it helps by expanding the datasets which will consequently improve the detection performance of the networks. The labelling was done using the Image Labeler App included in the Computer Vision System Toolbox 8.0 from Matlab [22]. We focused our efforts in gathering images that would add more diversity on situations that were not covered in the Corsican Dataset, mainly cases where the images are taken from long distances, from an aerial perspective and with small areas of fire/smoke.

Each dataset was randomly split into three separated sub-sets, one for training, one for validation and the last one for testing purposes (divided in the following proportions: 70%, 20% and 10%).
5.2 Network Training

As explained earlier, the system has in total four different network instances, each one having a different task. We must train two instances of the SqueezeNet and two instances of the U-Net. On the training stage it was used the training and validation sub-sets of each dataset. The models used in the system were previously detailed in Chapter 4.

5.2.1 SqueezeNet (Classification)

We firstly train the networks responsible for classifying if each patch contains fire or smoke using the classification dataset. We set the training parameters as follow:

**Fire Training Parameters:**
- **Optimizer:** Adam()
  - Learning Rate: 0.001
  - $\beta_1$: 0.9;
  - $\beta_2$: 0.999;
  - $\epsilon$: 1e-7;
- **Loss:** Binary Crossentropy;
- **Batch Size:** 32;
- **Patience:** 20;
- **Epochs:** 150;
- **Monitor:** Validation Loss;

**Smoke Training Parameters:**
- **Optimizer:** Adam()
  - Learning Rate: 0.001
  - $\beta_1$: 0.9;
  - $\beta_2$: 0.999;
  - $\epsilon$: 1e-7;
- **Loss:** Binary Crossentropy;
- **Batch Size:** 32;
- **Patience:** 20;
- **Epochs:** 150;
- **Monitor:** Validation Loss;

The parameters chosen were the ones to lead to the best losses while not adding too much processing time. The final training for fire took approximately two hours and a half and for smoke took two hours.

The progress of the model’s accuracy and loss (both in the training and validation datasets) are shown in Figure 5.2. Both graphs show a good evolution in the convergence, both reaching a steady line where, after reaching a patience number of epochs with no progress, the training is stopped.
At this point we can already make some assumptions about the differences of the performance between the fire and smoke classification. This can also be seen in Table 5.2, where the accuracy and loss values are shown, corresponding to the lowest validation loss point.

**Table 5.2: Training Performance Results**

<table>
<thead>
<tr>
<th></th>
<th>SqueezeNet Fire</th>
<th>SqueezeNet Smoke</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>Accuracy</td>
<td>0.9856</td>
</tr>
<tr>
<td></td>
<td>Loss</td>
<td>0.0411</td>
</tr>
<tr>
<td>Validation Set</td>
<td>Accuracy</td>
<td>0.9598</td>
</tr>
<tr>
<td></td>
<td>Loss</td>
<td>0.1436</td>
</tr>
</tbody>
</table>

The models have an overall good performance, although the smoke detection performance is a bit worst than the fire one. As expected, it is harder for the network to correctly train with a smaller dataset and the smoke phenomenon has more intricacies that make its detection quite more difficult than fire.
5.2.2 U-Net (Segmentation)

Similarly to the training stage of the two instances of the SqueezetNet, we now expose the training of the two U-Net’s. The parameters used in the training phase are as follow:

**Fire Training Parameters:**
- **Optimizer:** Adam()
  - Learning Rate: 0.001
  - $\beta_1$: 0.9;
  - $\beta_2$: 0.999;
  - $\epsilon$: 1e-7;
- **Loss:** Binary Crossentropy;
- **Batch Size:** 32;
- **Patience:** 30;
- **Epochs:** 200;
- **Monitor:** Validation Loss;

**Smoke Training Parameters:**
- **Optimizer:** Adam()
  - Learning Rate: 0.001
  - $\beta_1$: 0.9;
  - $\beta_2$: 0.999;
  - $\epsilon$: 1e-7;
- **Loss:** Binary Crossentropy;
- **Batch Size:** 32;
- **Patience:** 50;
- **Epochs:** 200;
- **Monitor:** Validation Loss;

The parameters used followed the same logic as before, trying to achieve the lowest value of loss for the validation dataset. The Adam optimizer proved again to be very capable of producing good results. The progress of the training stage for both the fire and smoke networks are shown in Figure 5.3, where we evaluate the progress’s losses and accuracy.
In Table 5.3 are shown the accuracy and loss values for the training point of lowest validation loss. Similarly as the classification stage, we again conclude that the smoke class has worst detection performance from the network due to same causes stated earlier.

<table>
<thead>
<tr>
<th></th>
<th>Training Set</th>
<th>Validation Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Loss</td>
</tr>
<tr>
<td>Fire</td>
<td>0.9241</td>
<td>0.0267</td>
</tr>
<tr>
<td>Smoke</td>
<td>0.9091</td>
<td>0.0682</td>
</tr>
</tbody>
</table>

Overall, the training results for the smoke class are considerably worst both in terms of accuracy and loss. On the U-Net dedicated to smoke there is a significant drop on pixel accuracy comparing the training set and the validation set results, contrary to the fire class, that proved to be more consistent.
5.3 Metrics

In order to correctly evaluate the results, it is important to choose a set of metrics that are generally used and are appropriate in this types of detection problems.

One of the most important metric tool is the confusion matrix. This matrix allows the visualization of the performance data from a detection algorithm, comparing the predictions made with the expected classes. An example of a confusion matrix for two classes is represented on Table 5.4.

Table 5.4: Example of a Confusion Matrix for two class detection

<table>
<thead>
<tr>
<th>Expected Classes</th>
<th>Fire</th>
<th>No fire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td>Fire</td>
<td>TP</td>
</tr>
<tr>
<td></td>
<td>No Fire</td>
<td>FP</td>
</tr>
<tr>
<td></td>
<td>Fire</td>
<td>FN</td>
</tr>
<tr>
<td></td>
<td>No Fire</td>
<td>TN</td>
</tr>
</tbody>
</table>

The True Positives (TP) and False Positives (FP) represent the number of correct and false detections respectively that were classified with positive class (fire in this case) by the algorithm. The True Negatives (TN) and False Negatives (FN) are similar, but this time for the negative predictions.

Using these values, a set of metrics can be calculated:

- **Precision:**
  
  \[
  \text{precision} = \frac{TP}{TP + FP} \tag{5.1}
  \]

  Precision effectively describes the purity of our positive detections relative to the ground truth. Of all of the objects that we predicted in a given image, it tells how many of those objects actually had a matching ground truth annotation.

- **Recall:**
  
  \[
  \text{recall} = \frac{TP}{TP + FN} \tag{5.2}
  \]

  Recall refers to the completeness of our positive predictions relative to the ground truth. From all of the objects annotated in our ground truth, it tells us how many did we capture as positive predictions.

- **Overall Accuracy:**
  
  \[
  \text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.3}
  \]

  The overall accuracy simply reports the percentage of pixels in the image which were correctly classified. This metric can sometimes provide misleading results when the class representation is small within the image, as the measure will be biased in mainly reporting how well you identify negative case. This correlates with our case, because the classes are unbalanced, the negative class represent a larger number of pixels than the fire or smoke class.
Therefore, the next alternative metric is better at dealing with this issue.

- Intersection over union (IoU):
  \[ IoU_{class} = \frac{TP}{TP + FP + FN} \]  

The Intersection-Over-Union (IoU), also known as the Jaccard Index, is one of the most commonly used metrics in semantic segmentation. The IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. This metric ranges from 0 to 1.

- Average IoU:

This metric consists of the average of all the different classes IoU values.

5.4 Tools Used

All the system components were developed using the set of hardware specifications and software tools we are going to mention next. The code was written and compiled inside the Google Colab virtualization environment using Python 3.6.9, running on a single core hyper threaded Xeon Processor at 2.3Ghz and a Tesla K80 (GPU). The libraries used for neural network development were Tensorflow 2.3.0 [23] and Keras 2.4.0. [24].

5.5 Experiments Catalog

A set of experiments were created in order to test the performance of the system as well as the performance of its respective individual components. As stated before, we will make use of the sub-dataset created specifically for the training phase.

On Table 5.5 we identify and describe all the experiments done in this work and on Chapter 6, the results produced by this set of experiments are displayed. To each experiment will be given an Id. to make it easier its identification and correlation with its respective results.
Table 5.5: Experiments Catalog

<table>
<thead>
<tr>
<th>Experiment Id.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment A.1.</td>
<td>Evaluation of the detection performance of the classification network for both fire and smoke to determine the best thresholds to be used.</td>
</tr>
<tr>
<td>Experiment A.2.</td>
<td>Similar study as Experiment A.1., but now for the segmentation network and its respective thresholds.</td>
</tr>
<tr>
<td>Experiment B</td>
<td>Experiment to study the influence of the patch size on the segmentation task (multi-scale problem).</td>
</tr>
<tr>
<td>Experiment C</td>
<td>Overall performance of the complete system vs. system with components missing.</td>
</tr>
<tr>
<td>Experiment D</td>
<td>Tests on images taken for the Firefront Project and on other aerial point of view images.</td>
</tr>
</tbody>
</table>

On Chapter 6, we will demonstrate the results produced within each different experiment in more detail.
## Results

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Experiment A.1.</td>
<td>41</td>
</tr>
<tr>
<td>6.2 Experiment A.2.</td>
<td>42</td>
</tr>
<tr>
<td>6.3 Experiment B</td>
<td>43</td>
</tr>
<tr>
<td>6.4 Experiment C</td>
<td>46</td>
</tr>
<tr>
<td>6.5 Experiment D</td>
<td>49</td>
</tr>
</tbody>
</table>
In this chapter, we will expose the experiment results produced while developing this thesis work. All results are identified with the corresponding Experiment Id. listed on the previous catalogue (Table 5.5).

### 6.1 Experiment A.1.

After the training stage of all the different networks, it is necessary to optimize a value for the corresponding detection threshold. Starting with the classification networks (SqueezeNet), we used the subset of the classification dataset reserved for test proposes to come up with the best threshold value. By doing a variation study, using the values of threshold ranging from 0.3 to 0.7, we determine the accuracy of class detection for both the fire and smoke networks. The results are shown on the Figure 6.1.

![Fire Classification Performance Variance](image1.png) ![Smoke Classification Performance Variance](image2.png)

**Figure 6.1:** Study on classification performance with different thresholds.

From this, we can conclude that the best value for the classification threshold for both networks is 0.4 or 0.5, both values corresponded to the highest accuracy values (0.900 for fire and 0.902 for smoke). Using threshold values different from 0.4 or 0.5 made the accuracy drop as we move away from them. Since the performance is the same in that range, the threshold value of 0.4 was selected and used for both fire and smoke classification networks.
6.2 Experiment A.2.

Likewise the previous experiment, the objective is to determine the best threshold value, but this time, for both segmentation networks. The test subset of the segmentation dataset is used to evaluate the performance of different thresholds using the average intersection over union metric. The performance variance is shown on figure 6.2.

![Fire Segmentation Performance Variance](image1)
![Smoke Segmentation Performance Variance](image2)

*Figure 6.2: Study on segmentation performance with different thresholds.*

The behaviour shown is very similar to the last experiment, but this time the value 0.4 results on a small advantage in performance against 0.5, on both networks. The variability of the threshold performance is more distinguishable because the metric is applied on a pixel level and consequently the sample size is a lot bigger than the study done for the classification networks.

From the smoke graph (Figure 6.2(b)), we can also state that the performance has a bigger variation then for the fire case. This also goes in favour of the assumption that smoke segmentation is less robust and that it is more difficult to bound the limits of the smoke regions.

The value for the segmentation threshold for both networks used in the system is consequently 0.4.
6.3 Experiment B

On this experiment we wanted to introduce the duality of the multi-scale problem that occurs when doing the segmentation. The system must be able to correctly segment large areas of fire/smoke while being able, at the same time, of doing a more precise and refine detection of smaller regions.

The input size of the U-Net is 128 by 128 (N), which is very small while compared to the dimensions of the dataset images. If we feed the all image into the network, very small areas of fire or smoke won’t be detected due to the lack of information caused by the image reduction. A way of dealing with this, is to slice the image into smaller patches and use them as the network input. Doing so, we retain more information contained on those images. However, the patch technique can lead to poor results on bigger areas of fire or smoke, because of missing external information from the patch that help define the phenomenon. Table 6.1 shows an evaluation using the classification and segmentation stages of the system on the test set in which we vary the patch size used to determine the Avg. IoU performance.

<table>
<thead>
<tr>
<th>Patch Size</th>
<th>Avg. IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xN</td>
<td>0.797</td>
</tr>
<tr>
<td>2xN</td>
<td>0.815</td>
</tr>
<tr>
<td>3xN</td>
<td>0.805</td>
</tr>
<tr>
<td>4xN</td>
<td>0.830</td>
</tr>
<tr>
<td>5xN</td>
<td>0.830</td>
</tr>
<tr>
<td>Full Image</td>
<td>0.849</td>
</tr>
</tbody>
</table>

The performance variability show that using the patch technique is not beneficial on a global perspective, as we use smaller patches the Average IoU is lower. This is due to the fact that the advantages of using patches do not overcome, in terms of Avg. IoU, the problems that occurs while detecting bigger areas. To better visualize this duality, we exposed on Figure 6.3 a case where using smaller patches is not beneficial. On the other hand, in Figure 6.4 is shown an example where feeding the network with smaller patches improves the detection performance.
As you can see in Figure 6.3(c), the patches inside the fire area were improperly labelled due to the lack of external information outside the patch that helps the networks to properly detect the fire. In contrast, in Figure 6.3(b), if the all image is used, the fire is correctly segmented.

Next, we introduce an example of the opposite aspect of this duality problem. The Figure 6.4 represents a case in which processing the image using patches lead to an increase of the detection performance. As stated earlier, these cases consist of images with very small portions of fire/smoke, like the one shown in Figure 6.4(b).
Looking at the detection results of the original image, we can see that using the patch technique improved the precision and the detail of the segmentation (shown in Figure 6.4(a)) while compared to the one using the full image as input. The very small areas of fire on the left side of the image are detected and the lower right region of fire is segmented with a relatively good level of detail in contrast of the results shown in Figure 6.4(c).

This problem motivated the creation of an algorithm that could dynamically decide what type of image patches to use in the detection to improve the system performance. This need was fulfilled with the QuadTree logic, explained on the previous chapter.
6.4 Experiment C

This experiment was used to evaluate the overall performance of the proposed system, and to see how it behaves with some of its components disabled. Once again we made use of the datasets reserved for testing proposes. The evaluation of each class is made separately and, for each, we tested four different models:

- **1. Q+C+S**: This model corresponds to the complete system created in this thesis work. It uses all the components explained earlier, the QuadTree algorithm, the classification stage and finally the segmentation stage;

- **2. Q+S**: This model is very similar to the first one, except that the component of classification is removed;

- **3. C+S**: On the next two models, the QuadTree methodology is removed and the input of the networks consists simply of the original images, there is no patch processing. This model includes the classification and segmentation stage;

- **4. S**: This final model consists only of the segmentation stage.

To evaluate the performance of the four different system models we use the Average IoU metric together with its Standard Deviation (SD). We also analysed the pixel accuracy and the average processing time per image.

Starting with the fire class, the results are shown on Table 6.2.

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. IoU</th>
<th>SD (Avg.IoU)</th>
<th>Pixel Acc.</th>
<th>Pross. Time p/ Img. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q+C+S</td>
<td>0.883</td>
<td>0.101</td>
<td>0.889</td>
<td>2.04</td>
</tr>
<tr>
<td>Q+S</td>
<td>0.828</td>
<td>0.150</td>
<td>0.888</td>
<td>1.52</td>
</tr>
<tr>
<td>C+S</td>
<td>0.849</td>
<td>0.131</td>
<td>0.884</td>
<td>0.23</td>
</tr>
<tr>
<td>S</td>
<td>0.849</td>
<td>0.131</td>
<td>0.884</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The complete systems showed a better performance while compared to the other models. Using the QuadTree methods, we achieved a considerable improvement from the model without it (improvement from 0.849 to 0.883 Avg.IoU). The Avg. IoU set of values were also less disperse while using the complete systems by comparing the SD(Avg.IoU) values. The results also show that, while using the QuadTree method, it really is beneficial to have the classification stage. If the QuadTree method is not used, the classification stage seems to not improve the performance, leading to belief that the segmentation is well capable by itself while using this dataset.
Regarding the processing time of each model, we noticed that using the complete system did not escalate on a large scale the processing time of each image. While compared with the simplest model 4, it took approximately ten times more.

We also did a more detailed study of the Avg. IoU distribution along all the dataset images to properly conclude the performance benefits. The results for all different models are shown in the box plot, on Figure 6.5.

![Box plot graphic of Avg. IoU distribution on fire segmentation](image)

**Figure 6.5:** Box plot graphic of Avg. IoU distribution on fire segmentation

As seen earlier on Table 6.2, the average value of IoU is considerably higher on the complete model, while the worst performance occurs on the system using the QuadTree algorithm with no patch classification before doing segmentation. The model 1 variation ranges in between slightly higher values and it was able to correctly detect the outlier case that was present on the other models results (result with approximately 0.5 as Avg. IoU).

From what we have seen so far, for the fire class, the complete system shows evidence of being a suitable and efficient solution for the multi-scale problem.
Next, the same evaluations are made, but this time for the smoke class, starting with the comparison of the four different models performance in terms of Avg.IoU, pixel accuracy and processing time. This comparison is shown in Table 6.3.

Table 6.3: Overall performance comparison of the complete system vs. system with missing components, on smoke segmentation

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. IoU</th>
<th>SD (Avg. IoU)</th>
<th>Pixel Acc.</th>
<th>Pross. Time p/ Img. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q+C+S</td>
<td>0.831</td>
<td>0.133</td>
<td>0.816</td>
<td>1.91</td>
</tr>
<tr>
<td>Q+S</td>
<td>0.772</td>
<td>0.149</td>
<td>0.794</td>
<td>1.43</td>
</tr>
<tr>
<td>C+S</td>
<td>0.818</td>
<td>0.134</td>
<td>0.799</td>
<td>0.22</td>
</tr>
<tr>
<td>S</td>
<td>0.818</td>
<td>0.134</td>
<td>0.799</td>
<td>0.19</td>
</tr>
</tbody>
</table>

In general, the four model performances behaved very similarly to the fire class. Again, the complete system has the best detection performance. Although the standard deviation is quite similar to the other models, this time there was a more significant improvement on pixel accuracy. The processing time escalation was very similar to the fire class and, once again, the worst model was the Q + S model, reinforcing the assumption that the QuadTree methodology does in fact require the classification of each patch in order to become an advantageous model.

Similarly as before, we represent in Figure 6.6 the box plot of the average IoU variation within the smoke class results.

![Box Plot of Avg. IoU Distribution](image)

**Figure 6.6:** Box plot graphic of Avg. IoU distribution on smoke segmentation

From this box plot, the poor performance of model 2. is quite more evident, the IoU distribution is more disperse and shows a lot of results near the 0.5 value. The complete system model shows a better median and a less scattered distribution although it does not reach the maximum values registered on models 3 and 4.
Also, every model shown the presence of an outlier.

After this experiment, once again, we proved that the smoke detection task is a lot more challenging. The results for the fire class are generally more consistent and the resulting segmentations are more in line with the ground-truth masks of the dataset.

6.5 Experiment D

This experiment has the purpose of testing the system with images that better represent the real life scenarios in which the system will be used. We gathered a set of images taken from a aerial point of view from a considerable distance, simulating the perspective of the vehicle that would take the wildfire images. Some of the images belong to a dataset specially collected for works related with the Firefront Project and others were collected from the web.

On Figure 6.7, we shown two results from the Firefront dataset in which the fire areas are very small (to better visualise the results, we added a zoom crop in both cases).

![Image](a) Fire Image 1 on the left, resulting detection on the right and zoom crops in the middle

![Image](b) Fire Image 2 on the left, resulting detection on the right and zoom crops in the middle

**Figure 6.7:** Fire tests on Firefront Dataset images
These types of images, where the aerial vehicle is quite far from the fire, represent a challenge for networks with a small input layer. For that reason, we developed the QuadTree method. The method was shown to be very capable and the resulting segmentations display a high level of detail and precision.

The same level of performance can be seen in Figure 6.8, where we test more images taken from a long distance to the wildfire.

![Small fire area image 1](image1.jpg) ![Resulting detection from image 1](image1_detection.jpg)

![Small fire area image 2](image2.jpg) ![Resulting detection from image 2](image2_detection.jpg)

**Figure 6.8:** Tests on other images taken from a long distance

Next, we make similar tests for the smoke class. In Figure 6.9 we tested one of the images from the Firefront dataset to see how the smoke detection behaves.
Looking at the figure, we can see that the resulting segmentation is not as precise as we have seen for the fire class. The boundaries that limit the smoke area are not as well defined and regions that appear to have similarities with smoke are sometimes incorrectly detected as being smoke (in the figure, the smoke extends along one of the dirt roads).

Similarly, on Figure 6.10, we can see the same type of behavior.

The smoke region is correctly detected and the background clouds are avoided although there are some false positives near the lake and the left region of the image. In spite of occasional poor detections that were detected, in general, the system dedicated for smoke detection is able to segment with a relatively good level of accuracy. At more extent, we showed on Figure 6.11, examples of well performed detections.
The results achieved in this experiment, support the concluding statements made throughout this work. The system dedicated for the fire class has better performance overall and the resulting detections are more in-line with the original image labels. The boundaries created seem to be correct and very precise.

The system developed for smoke detection shows a performance decrease while compared to the fire one but, in general, the system is well capable of detecting even small smoke regions. The detections produced tend to be less precise and the limits are more vague and ambiguous. As stated earlier, smoke detection is a lot more challenging and, consequently, this is the component that could be more significantly improved in future works.
Conclusions
As we reach the end of this thesis work, some conclusions can be drawn about the overall experience, results and possible future work.

The system developed shows some innovative characteristics that distinguish him from other state-of-the-art methods and techniques mentioned earlier. The system makes use of the powerful capabilities of CNNs, detecting and segmenting with a high level of precision both the fire and smoke regions while being very adaptable and computationally efficient. The proposed methodology also addresses the important multi-scale problem that comes associated with detections from a long distance to the object, a topic that is not addressed by any approach from the state-of-the-art that we have studied. The association of the best state-of-the-art CNNs with the original QuadTree methodology form a very complete and reliable system that comply with the objectives of this thesis project.

The overall performance of the proposed systems is highly dependent on the behavior of the neural networks used. The training phase of said networks is a really important factor and, consequently, the datasets used should be as complete and extensive as possible. From the research it was made, there is a lack of datasets with fully segmented labelled images for both the fire and smoke classes which made this work more challenging. If there was a centralized, open-source, complete and fully labelled database of fire and smoke images that would have been be a really significant improvement step in this research area. Working with bigger and better datasets could have really helped to improve the results achieved in this work, specially on the smoke detection task, as this shown to be a very challenging factor.

Due to the small size of the dataset gathered at early stages, we decide to manually create labeled images to add to our dataset. This type of tasks are really demanding and time consuming but they usually pay off, as we increased the performance of the networks used. This task is hugely subjective. Two different subjects are going to label the images differently, specially the smoke class. This fact can be an interesting topic of study, to find how much a subjective labelled dataset can influence a detection problem and to quantify the differences of the labelled images among different people.

In this work, only RGB images are used to do the fire detection. In future studies it would be interesting to use other different types of cameras (for example thermal and infra-red cameras) while trying to fuse the different spectral image information to see if it improves the performance of the detection. Using other spectres can be a demanding task, due to the complete lack of datasets of non-RGB images of wildfires. By only using RGB images, our system is limited by the atmospheric conditions. If smoke or clouds are covering the wildfire areas beneath the vehicle, no fire will be detected. Solving these challenges could represent a future progress step of this work.

Also as progress to this work, after doing the fire detection, it would be interesting to have a component that would outline the fire front in shape of elongated lines. This would represent a connection link between the fire detection task and other project components, for example the geo-referencing of the fire or components related with the fire behaviour prediction.
Finally, looking at the results produced by the developed system, we can conclude with high level of confidence that the system is capable of achieving the proposed objectives in which this thesis work was based on. From the tests done on real images taken from an aerial point of view, the systems shown to be capable of detecting both fire and smoke with a relatively low time and computational cost, making it a truly viable option of an implementation that could help the brave fire fighting teams.

The project developed in this thesis was also published on a Portuguese scientific journal [25] related with pattern recognition, proving the merit and the quality of the work produced. The publication of the article and the achieved results help to confirm the true potential of the proposed methods.

We truly hope that the work produced in this thesis project represent a powerful addiction in the Firefront Project and that it could help to motivate other individuals to study this interesting topic.
Bibliography


The code developed in this thesis work was published into GitHub. The code can be reviewed using the following link:

https://github.com/g0nzal0rd/FireFront-Detect

The network implementations are based on the codes as follows:
SqueezeNet [26]
U-Net [27]