TÉCNICO LISBOA

Fire and Smoke detection with weakly supervised methods.

Bernardo Filipe Morais Amaral

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Ana Catarina Fidalgo Barata
Prof. Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Ana Catarina Fidalgo Barata
Member of the Committee: Prof. Luis Filipe Barbosa de Almeida Alexandre

September 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.

Acknowledgments

First, I would like to thank my supervisors, Prof. Alexandre Bernardino and Prof. Catarina Barata, for the opportunity, guidance and all the valuable insights during this time. Their patience, accessibility and collaboration have made this work possible. I also want to thank Milad Niknejad that helped in the project development.

To my family who have always supported and encouraged me to do more and better during these years away from home. A special thanks to my mother who was always a source of motivation and support, to my father who ensured that nothing was ever missing. A big and special thanks to my twin sister for always being tremendously helpful and without whom it would not have been the same. To my grandfathers for their outstanding support and help. A heartfelt thanks to my uncle who left us too soon, but who always taught me valuable lessons.

I would like to thank all my friends with whom I shared this adventure. A special thanks to the friends I made throughout this journey in Instituto Superior Técnico. Not only did they help with the study and the projects, they made this an incredible journey. A special thanks to my friends in my city that have been with me for a long time and are always there at the weekends to do a break. Last but not least, a special thanks to the FSTLisboa team, especially to the entire Autonomous Systems department, for the unbelievable challenge that was to build an autonomous race car.

This work was made in the context of the FCT project FIREFRONT (PCIF/SSI/0096/2017).

Abstract

Forest fires have become a recurring disaster worldwide and, every year, thousands of hectares of forests are devastated. The impacts on nature and society are disastrous. To develop robust deep learning methods for fire and smoke detection a large number of data and the related annotations are required. However, the number of publicly available forest fires datasets is very scarce. For this reason, this work proposes an alternative system capable of detecting and localizing fire and smoke in aerial images using only weakly-supervised deep learning methods. A classification model was trained using only image-level labels and, from there, the information from the convolutional layers was extracted to create the first iteration of a segmentation mask. Afterwards, by combining it with the colour and spatial information of the original image, one can create a segmentation mask that can correctly detect the fire/smoke zones. The proposed method was tested and proven to be able to accurately detect fire/smoke at the pixel-level despite never being trained with any supervision at that level. Compared with other fully-supervised methods, the results show that when considering their heavy needs, the proposed weakly-supervised system can strongly compete with them.

Keywords

fire detection, smoke detection, convolutional neural networks, weakly-supervised methods, deep learning

Resumo

Os incêndios florestais tornaram-se um desastre recorrente em todo o mundo e, todos os anos, milhares de hectares de florestas são devastados. Os impactos na natureza e na sociedade são desastrosos. Para desenvolver métodos robustos de aprendizagem profunda é necessário um grande número de dados e respectivas anotações. No entanto, o número de datasets sobre incêndios florestais disponíveis publicamente é muito escasso. Por esta razão, este trabalho propõe um sistema alternativo capaz de detectar e localizar o fogo e fumo em imagens aéreas, utilizando apenas métodos de aprendizagem profunda com fraca supervisão. Foi treinado um modelo de classificação utilizando apenas anotações de nível de imagem e, a partir daí a informação das camadas convolucionais foi extraída para criar a primeira iteração de uma máscara de segmentação. Posteriormente, ao combiná-la com a cor e informação espacial da imagem original, consegue se criar uma máscara de segmentação que detecta correctamente as zonas de fogo/fumo. O método proposto foi testado e provou ser capaz de detectar com precisão o fogo/fumo ao nível dos pixel, apesar de nunca ter sido treinado com qualquer supervisão a esse nível. Em comparação com outros métodos de segmentação fortemente supervisionados, os resultados mostram que, ao considerar as suas elevadas necessidades, o sistema proposto de fraca supervisão consegue competir com eles.

Palavras Chave

detecção de incêndios, detecção de fumo, redes neurais convolucionais, métodos fracamente supervisionados, aprendizagem profunda

Contents

List of Figures

List of Tables

Acronyms

CRF	Conditional Random Fields
FCCRF	Fully Connected Conditional Random Fields
CAM	Class Activation Mapping
GT	ground-truth
CNN	Convolutional Neural Network
GAP	global average pooling
ANN	Artificial neural network
AN	artificial neuron
ReLU	Rectified Linear Unit
MLP	Multi-layer Perceptron
SGD	Stochastic Gradient Descent
RGB	red green blue
TP	true positive
FP	false positive
FN	false negative
TN	true negative
IoU	intersection over union
mIoU	mean intersection over union
IQR	interquartile range
UAV	unmanned aerial vehicle
WSS	Weakly Supervised Segmentation

1

Introduction

Contents

This thesis presents a study for the implementation of a system in aerial vehicles to detect fire and smoke zones in order help the firefight teams to create a wiser and faster firefight. It intends to do so only using methods that do not rely on strong supervision since the lack of data in this field makes the development of robust algorithm a great challenge.

This work is related to the Firefront project and aims to a develop system to assist fighting forest fire by providing information on real-time of the localization of fire and smoke. By providing to the teams an automated method to detect fire and smoke in which they can rely, the process of approaching a fire can change significantly. A good localization and the respective evolution of the fire can drastically prevent damages. The fire must be fight as soon as possible since if it takes large proportions it can become uncontrollable and so, the early detection of smoke columns takes an important role to detect a forest fire in its' early stages.

1.1 Motivation

Forest fires are a scourge that every year destroy thousands of hectares of forest around the world. They have a series of effects on both the burned area and the underlying areas. The consequences go beyond the visible effects on nature and society such as the destruction of material assets and the effect on vegetation. The entire ecosystem is threatened, from fauna and flora with loss of biodiversity, soil degradation and erosion, to CO_2 emissions. According to [3], in Portugal between 2017 and 2019 about 630 thousand hectares were burnt due to forest fires. Only in 2017 more than 100 people have died. Thus, to mitigate the dangers and minimize the impacts on people and nature, there is the need to have systems capable of doing effective prevention, early warning, and a clever firefight.

Traditionally, forest fires were mainly detected by human observation from lookout towers. This can be very limited, inefficient and are still be subject to human error. After the detection, it is up to the firefighting units to evaluate and study how the firefight will be carried out. For this, they strongly depend on the information about the fire site and its environment. So, the earliest the fire is detected, the more effective the firefight will be. Therefore, it is important to detect the fire and its fire fronts in their early stages to anticipate the behaviour over time and especially the position and intensity However, in its great majority, this information is scarce, depending sometimes on the perception of the agents involved in the field which can be very limited.

The use of vision-based systems can transform the traditional methods into a reliable process providing as much information as possible. There is a wide range of options from terrestrial systems through the use of surveillance systems, to aerial systems that rely on the vision sensors in aerial vehicles or even to spaceborne systems that use satellites. The one that can cover a greater amount of useful information for firefighters will undoubtedly be the use of vision-based systems in aerial vehicles since having real-time visual information from the fire site could be of extreme value for the combat teams.

The Firefront project [1] intends to develop a system to help and support the firefighting teams. This solution consists in creating a much efficient and fast firefight by detecting and tracking fire fronts and their possible reburns. This process would be carried out through the use of manned vehicles or unmanned aerial vehicles (UAVs) that are equipped with red green blue (RGB) and infrared cameras and other sensors and communication systems. These vehicles will fly over the affected areas, sending the information collected to a ground station in real-time. This ground station will process all the information and will then transmit the detection of the fire and smoke zones and their respective georeferenced coordinates. Then, with the addition of other meteorological factors such as wind speed and direction, a prediction of the fire fronts evolution can be made.

Early detection and monitoring of fire outbreaks and fronts are crucial as once a forest fire reaches a certain size it can be hardly controlled. Also, the detection and analysis of smoke columns are essential to detect early stages of fire and define the type of fire and the direction it may take. All this information is of great importance for the fire fighting teams and can greatly contribute to a faster and wiser fire fighting.

The detection phase could benefit from the recent and strong advances in computer vision and machine and deep learning. These techniques have already proven to be capable of tackle several complex problems. By combining these techniques with vision-based systems in aerial vehicles, it is possible to detect fire zones in a more automated and robust way.

1.2 Challenges

The detection of fire and smoke through an image-based system poses a considerable challenge since neither fire nor smoke has a well-defined shape and a constant colour. The usual methods of locating/identifying objects using deep learning are trained on a large amount of fully annotated data, which means that in each image of the dataset there must be an annotation of where each class is present in the image, for example through the use of bounding boxes or pixel-level annotations. Though, the creation of such annotations is very expensive and, especially in the case of fire and smoke, it can be very subjective, since they can take very irregular shapes. Moreover, a single annotator could introduce some bias in the annotation which limits the models capacity to generalize well on new data.

To demonstrate the cost and possible ambiguity of creating labels at the pixel level by hand, an experiment was conducted in which 10 different people were asked to annotate 3 images of fire and 3 images of smoke. The idea behind this task was to have multiple annotators for the same set of images and understand how each one separates the fire and smoke zones. Also, we wanted to understand how much time each annotator takes to label each image.

Figure 1.1 and 1.2 illustrate the result of the experiment for two images of fire and smoke respectively

[1] http://www.firefront.pt/

from three different random annotators. For fire, the major difference between the annotations is the level of detail each annotator puts in the mask since the shape of fire can be very irregular. This results in a difficult process and consequently a large time to create the mask. For smoke , the issue was with zones where the smoke is very dim or where it not clear if its smoke or clouds. Some annotators were more conservative, delimiting only zones where smoke was very dense and others decide to delimit it by excess.

In both cases, the differences on the annotations by different annotators are very clear. This demonstrate how ambiguous and expensive the annotations can be.

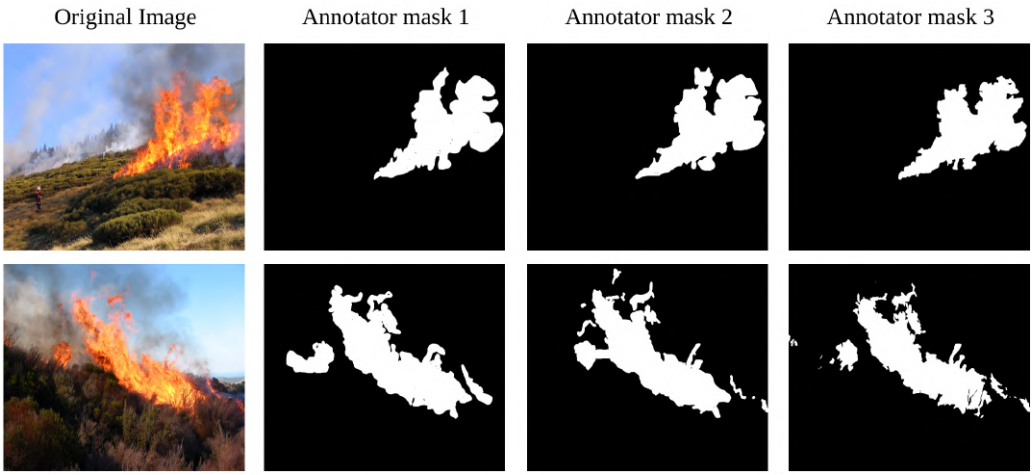


Figure 1.1: Fire examples different annotations

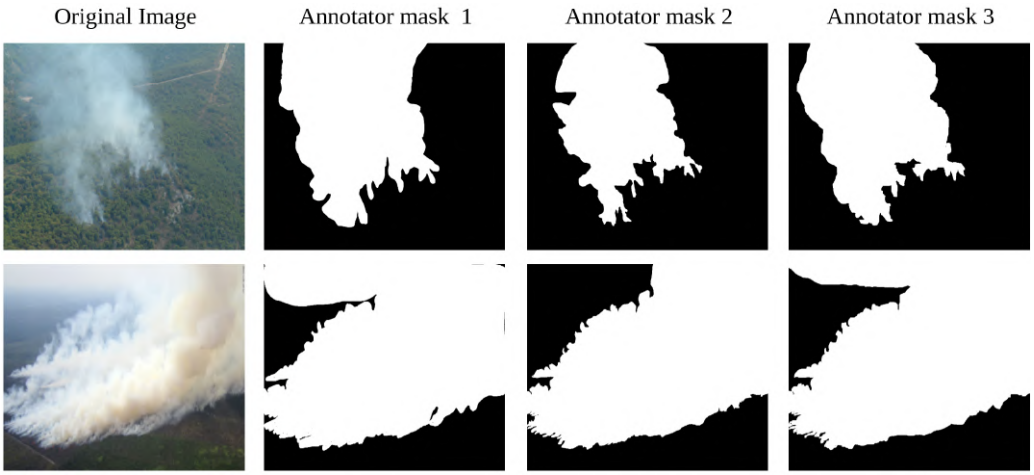


Figure 1.2: Smoke examples different annotations

According to this small experiment with 6 images, each label took on average 6 min and 41 sec. For a dataset of 2000 images, it would take 13366 minutes and 40 seconds. This is equivalent to 223 hours or approximately 10 full days of work.

A possible alternative to this complex and time-consuming process is the use of image-level annotations. In this type of annotation, instead of having information about the class in each pixel, there is only information about the presence of that class in the entire image as in Figure 1.3. This way, it is possible to create a greater number of annotations in a short period of time. However, this type of annotation has the cost of losing detail in the annotation. It is then necessary to understand whether the advantage of getting a greater number of annotations outweighs the cost of losing detail in the annotations.

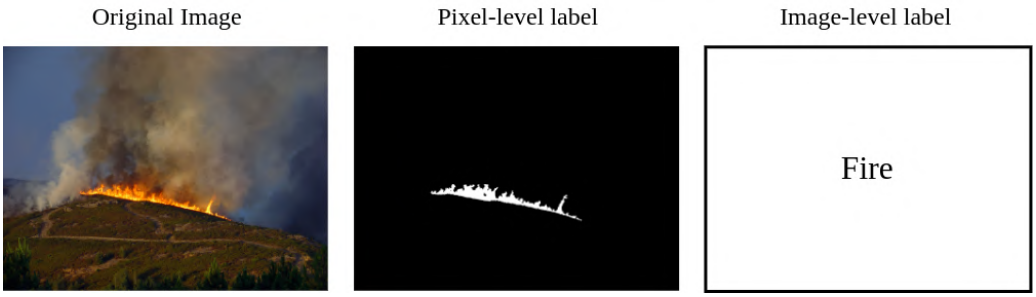


Figure 1.3: Labelling methods

On the other side, the creation of an image-level label can easily be done through visual inspection in just a few seconds. Considering by excess that each image-level label takes about 10 seconds to create and comparing it with the average time that the annotators took to create the pixel-level labels, it can be concluded that the the creation of a pixel-level annotation takes about 50 times more time than an image-level annotation.

1.3 Objectives

This thesis addresses a possible solution for the implementation of a system capable of detecting fire and smoke using images taken by aerial vehicles.

Traditional methods of object detection and segmentation through deep learning require large amounts of images and their pixel-level annotations. However, in the area of forest fires, this type of data is quite scarce. On the other hand, it is easy to collect examples from online images but the creation of pixel-level labels is extremely expensive. Therefore, it is aimed to use the few existing datasets in this study area and complement them with as many images as possible. With this it is intended to create a method for fire and smoke segmentation using only annotations at the image-level. So, we will use models that will only be trained using image-level annotations. This way, it is expected that the deep learning model will learn the features of the class present in the image and be able to produce segmentation masks without ever being trained with the information that is typically used for this purpose which are annotations at the pixel level.

In the end it is intended to prove that the use of this type of methods for fire and smoke detection

is advantageous when compared with the expensive process of creating pixel-level annotations. The final objective is then the creation of segmentation masks that can correctly identify fire and smoke zones at the pixel level.

1.4 Thesis Outline

This thesis is is organized as follows:

- Chapter 2 presents a review of the background knowledge about convolutional neural networks, image classification and weakly supervised learning. Moreover, it presents the state-of-the-art about fire and smoke detection and weakly supervised object detection.
- Chapter 3 presents the proposed work and exposes the methodology of all the stages used in the proposal.
- Chapter 4 presents the datasets used, makes the connection between the methods to use and the problem to solve and finally it sets a list of experiments to evaluate and validate the proposed approach.
- Chapter 5 presents the results for the experiments and evaluate the overall system performance.
- Chapter 6 outlines the most relevant parts of the work and list some suggestion for future work in order to make some improvements.

2

Background and State of the Art

Contents

This chapter performs a literature review on fire and smoke detection state of the art. The methods for the detection can be divided into classic methods which rely on basic components of fire and smoke images, as their colour components and texture, or deep learning methods that rely on neural networks to extract and understand the features that characterize fire and smoke. For the latter ones, it starts by doing a background introduction on key deep learning concepts and generic methods related to those networks. Then it is introduced and compared two deep learning methods according to the type of annotations used for the model training. Finally, it is introduced significant works in the field of fire and smoke detection using the two types of methods described and using different systems and sensors.

2.1 Background

2.1.1 Neural Networks

Artificial neural networks (ANNs) were originally inspired by the incredible functionally of the human brain to analyze and process information. Similarly to the human brain, an ANN is formed by a group of neurons interconnected, in this case, artificial neurons. The structure of an artificial neuron (AN) is illustrated in Figure 2.1. It receives inputs $x = (x_1, x_2, ..., x_n)$ from several other ANs, multiplies them by respectively assigned weights w_i, adds them plus a bias value θ and finally applies an activation function F to obtain the output y. Mathematically it can be translated through Equations (2.2) (2.1).

$$y = F(\xi(x, w)) \tag{2.1}$$

$$\xi(x, w) = \theta + \sum_{i=1}^{n} w_i x_i \tag{2.2}$$

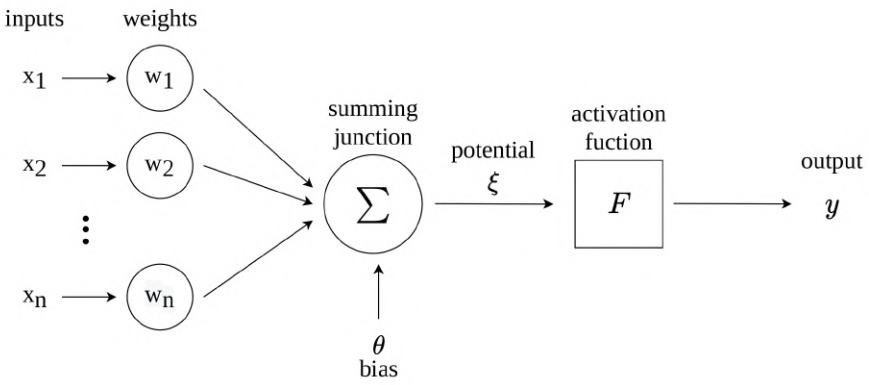


Figure 2.1: Structure of a Neuron

In 1958, *Frank Rosenblatt* introduced the *perceptron* in [4] as an ANN with a single layer of neurons.

A Multi-layer Perceptron (MLP) is, as the name suggests, an ANN with multiple layers as illustrated in Figure 2.2. The first layer is called Input Layer since contains the data for the model. Each neuron here represents a unique attribute in the data. The intermediate layers are called hidden-layers and they are responsible for applying multiple transformations on the inputs and thus extracting representative features. Usually, there are several of them and they are typically fully-connected which means that each neuron receives input from all the previous layers' neurons. The features are then passed to the output layer which is then responsible to return an output representing the model prediction.

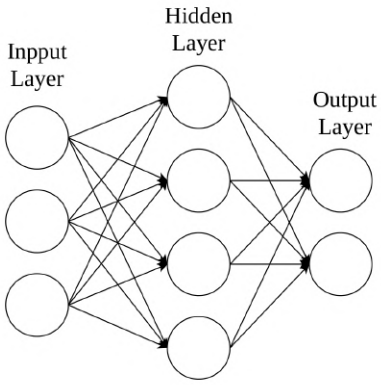


Figure 2.2: Example of a MLP

The Activation functions are responsible for modifying the input before passing it to the next layer. They can be divided into linear and non-linear. The linear ones are very limited to simple situations with a straight relationship between the input and the outputs. The non-linear ones give the ANN their potential, allowing to model non-linear complex relationships. There are three proprieties that helps to create an ANN. First, the non-linearity to model highly complex relationships. Second, being continuously differentiable so that the output has a nice slope and it is possible to compute error derivatives with respect to the weights and understand which direction to update the weights in the training process . Third, to have a fixed range, so that the input data is compressed into a range that makes the model more stable and efficient. The most used activation functions [5] are:

- **Step** - Transform the input into binary values and it is non-linear.

$$f(x) = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x \geqslant 0 \end{cases} \tag{2.3}$$

- **Rectified Linear Unit (ReLU)** - The most used activation function in hidden-layers as it is non-linear. It returns the value provided as input directly or 0 if the input is smaller or equal to 0.

$$f(x) = max(0, x) = \begin{cases} 0 & if\ x < 0 \\ x & if\ x \geqslant 0 \end{cases} \tag{2.4}$$

- **Hyperbolic Tangent** - Transforms the input into the range of -1 and 1. Commonly used in hidden-layers since its output is zero-centred, is non-linear and is continuously differentiable.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.5}$$

- **Sigmoid** - Transforms the input into the range of 0 and 1. Commonly used as an output when one wants to predict the probability of each final neuron independently. It is non-linear and is continuously differentiable.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

- **Softmax** - Similar to Sigmoid but takes into account the other neurons in the layer to calculate a normalized probability. Where x_i represents the input of the neuron, x_j the inputs of the remaining ones and K the number of neurons.

$$f(x) = \frac{e^{x_i}}{\sum_{j=0}^{K} e^{x_j}} \tag{2.7}$$

Typically, the weights of an ANN are initialized randomly. The process of training an ANN consists of iteratively update the weights that connect each layer by minimizing a loss function. The objective is to determine the importance of each neuron to the output. A loss function indicates the performance of the model to make predictions with the current set of weights by evaluating the error between the outputted predictions $\widehat{y}$ and the ground-truth results y. Several losses can be used depending on the task the ANN will be trained for. The most commonly used are:

- **Mean Absolute Error** - Used for regression problems. Measures the average magnitude of the errors between the predicted and ground-truth values without considering their direction. Where n represent the number of classes.

$$J(y, \widehat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \widehat{y_i}| \tag{2.8}$$

- **Mean Square Error** - Very similar to the previous one but now summing the quadratic error.

$$J(y, \widehat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y_i})^2 \tag{2.9}$$

- **Binary Cross-Entropy** - Used for binary classifications whit exactly 2 classes. Measure the difference between the actual and predicted probability distributions.

$$J(y, \widehat{y}) = -y_i \, log(\widehat{y_i}) + (1 - y_i) \, log(1 - \widehat{y_i}) \tag{2.10}$$

- **Categorical Cross-Entropy** - Similar to Binary Cross-Entropy but for a multi-class classification when the number of classes is greater than 2. Measure the average difference between the actual and predicted probability distributions for all classes. Where n represent the number of classes.

$$J(y, \widehat{y}) = -\sum_{i=1}^{n} y_i \, log(\widehat{y_i}) \tag{2.11}$$

The optimizers aims to iteratively solve the minimisation of the loss function in the training process. They dictate how the weights will be updated in each iteration so that the error between prediction and ground truth turns as small as possible. The most commonly used are listed below. The J is the loss function, $w^{(t)}$ is the set of weights at instance t and $w^{(t+1)}$ at instance $t+1$, ∇_w is the gradient relative to the weights [6].

- **Gradient Descent** - Updates all the weights in the opposite direction of the gradient of the loss function. Here η is the learning rate and sets the size of the steps it takes to reach the local minimum. The learning rate must be set carefully in order not to diverge.

$$w^{(t+1)} = w^{(t)} - \eta \ \nabla_w J(w^{(t)}) \tag{2.12}$$

- **Stochastic Gradient Descent (SGD)** - Similar to Gradient Descent but only updates the weights for each training example y_i and $\widehat{y_i}$ selected randomly, decreasing the time of convergence and the high computational cost.

$$w^{(t+1)} = w^{(t)} - \eta \ \nabla_w \, J(w^{(t)}, y_i, \widehat{y_i}) \tag{2.13}$$

- **SGD with momentum** - The momentum helps to accelerate the SGD in the relevant direction and dampens oscillations by adding a fraction α of the update vector of the past time step to the current update.

$$v^{(t+1)} = \alpha \, v^{(t)} + \eta \, \nabla_w J(w^{(t)}, y_i, \widehat{y_i}) \tag{2.14}$$

$$w^{(t+1)} = w^{(t)} - v^{(t+1)} \tag{2.15}$$

- **Mini-batch gradient descent** - it performs the update for every mini-batch of training examples $y_{i:i+n}$ and $\widehat{y}_{i:i+n}$. It combines the good convergence of the normal Gradient Descent with the low time of convergence of the SGD. It can be used with momentum as well. It is similar, in its normal version to Equation (2.12) and in momentum version with Equation (2.14), with the only difference being that loss function is $J(w^{(t)}, y_{i:i+n}, \widehat{y}_{i:i+n})$.

- **Adam** - Updates the weights through a momentum by using an exponentially decaying average of past gradients $m^{(t)}$ and past squared gradients $v^{(t)}$. Where β_1 and β_2 are reals close to 1. So, $m^{(t)}$

and $v^{(t)}$ are estimates of the first and the second moment of the gradients, respectively.

$$g^{(t)} = \nabla_w J(w^{(t)}) \tag{2.16}$$

$$m^{(t)} = \beta_1 \, m^{(t-1)} + (1 - \beta_1) \; g^{(t)} \tag{2.17}$$

$$v^{(t)} = \beta_2 \, v^{(t-1)} + (1 - \beta_2) \; g^{(t)^2} \tag{2.18}$$

In order to counteract the zero bias, one computes the bias-corrected first and second moment estimates:

$$\widehat{m}^{(t)} = \frac{m^{(t)}}{1 - \beta_1^{(t)}} \tag{2.19}$$

$$\widehat{v}^{(t)} = \frac{v^{(t)}}{1 - \beta_2^{(t)}} \tag{2.20}$$

Finally, they can be used to update the weight.

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta \; \widehat{m}^{(t)}}{\sqrt{\widehat{v}^{(t)}} + \epsilon} \tag{2.21}$$

The weights are then changed according to the how the loss function is evolving during the training phase.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of ANN that recognize patterns in data through convolutional operations [7]. They are commonly associated with computer vision problems since they can easily model spatial patterns in images. A typical classification CNN is illustrated in Figure 2.3 and can be divided into two parts, the backbone and the classifier. The backbone is the feature extractor of the network, which is responsible for computing the features from the input data using a series of convolutional layers. Then, the features are fed to the classifier, which in turn is responsible for generating an output through the use of fully-connected layers.

A convolution is an element-wise matrix multiplication where one of the matrices is the image, and the other is the kernel that transforms the image into a convolved feature. In Figure 2.4 the kernel is illustrated in the middle. The kernel slides into all the parts in the image to perform the convolution. Each operation generates a new pixel in the output tensor. The stride defines the size of the step the kernel takes between operations. Figure 2.4 illustrates a convolution operation between an image with dimensions 5 x 5 x 1 and a kernel with dimensions 3 x 3 x 1 and stride equal to 1. That process results in a convolved feature with dimensions equal to ($M_{rows\ Image} - M_{rows\ Kernel} + 1$) x ($N_{columns\ Image} - N_{columns\ Kernel} + 1$) x $C_{channels\ Kernel}$.

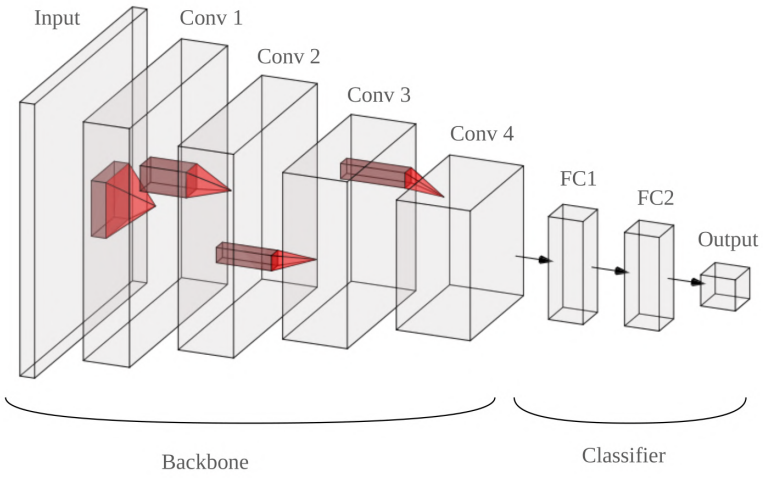


Figure 2.3: Example of CNN

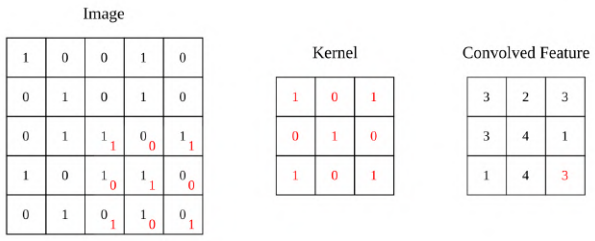


Figure 2.4: Example of a convolution operation

By default, the filter will start in the top left corner of the image and so the borders of the image will only be visited once while the remaining pixels will be visited multiple times. To attenuate this border effect, the image can suffer padding before the convolution where a set of zeros is added all around the image. This way and keeping a stride equal to 1, the dimensions of the convolved feature are the same as the image. Figure 2.5 illustrates this process. So, a convolutional layer is then a layer responsible for performing all the convolutional operations in the image.

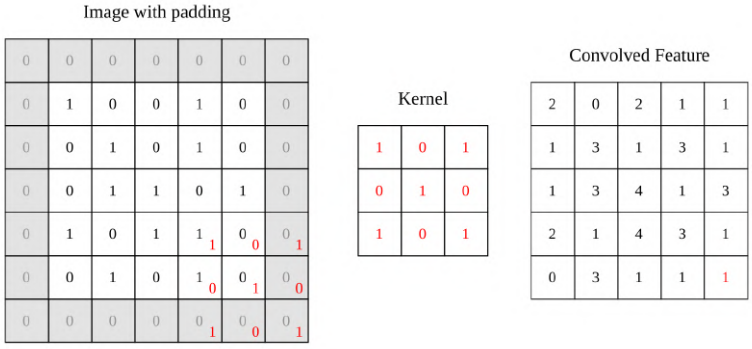


Figure 2.5: Padding

The pooling layers performs downsampling in feature maps by summarizing the presence of features in patches. This way, the most relevant features are extracted with invariance to local translations. The pooling can be applied using filters with fixed dimensions (M *rows* by N *columns*) and a stride value, similarly to a convolution layer, or be applied in full to the whole image performing a global pooling. There are two types of pooling, the Max and the Average Pooling. Max Pooling returns the maximum value from the patch of the image covered while the Average Pooling returns the average of them. Figure 2.6 illustrates the two types of pooling using a 2 x 2 filter and a stride of 2.

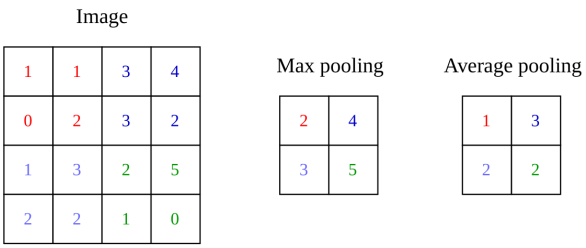


Figure 2.6: Pooling

2.1.3 Weakly Supervised Learning

Deep convolutional neural networks have achieved great success in several computer vision tasks. Their performance is directly related to the quantity and quality of the data the networks are trained with. The best performance is achieved on fully-supervised methods where it is used the most informative supervisory data possible. In a segmentation problem, the use of fully supervised data means the use of pixel-level labels. This type of labels work as an attention mechanism since only relevant parts are retained for analysis and all irrelevant parts of the image are ignored. However, collecting large-scale accurate pixel-level annotation is very time-consuming requiring expensive human effort. Therefore, weakly supervised methods try to perform the same computer vision tasks but only using weakly annotated data. So, weakly supervised segmentation aims to create segmentation masks with only image-level labels. Unlabeled and weakly-labelled visual data can be collected in large amounts in a relatively fast and cheap manner. Also, the process of labelling the unlabeled data at the image level is almost effortless.

2.2 State of the Art

This section is divided in two parts: subsection 2.2.1 highlights relevant works on weakly supervised segmentation methods and subsection 2.2.2 reviews the current state of the art in fire and smoke detection using classic and deep learning methods.

2.2.1 Weakly Supervised Segmentation

Most of the weakly supervised segmentation approaches are based on the Class Activation Mapping (CAM) method [1], which will be detailed in Section 3, since it will also be an important component of this work. From this starting point, various techniques can be used either to improve this approach by making it more robust and accurate or integrating it with other methods.

When doing a Weakly Supervised Segmentation (WSS) the output mask is obtained using the features in the image that the classification model used to make the prediction. Sometimes, these features will only contain the most relevant and distinguishable parts of the object. Several authors proposed different approaches to attenuate this issue and create a mask containing the whole extension of the object to detect and not only its most discriminative part. On [8] they proposed to use a Hide-and-Seek approach where they randomly occlude patches in the training images so that each image can be used for training multiple times but with none or different patches occluded. This way they force the network to learn different features that characterize an object. Also using Hide-and-Seek, the authors of [9] train a siamese network where one of its sides only uses the normal images and the other side used the images with the patches occluded. On [10] an adversarial complementary learning approach is proposed. After the network backbone, they divide it into two branches. The first branch selects the most discriminative feature maps and feeds them to the classifier. On the second branch, chosen feature maps from the first branch are removed and the remaining feature maps are feed to the classifier. By doing so, they force the classifier to use less relevant features to classify the object. The final mas will be a combination of the selected feature maps from both branches.

Instead of performing architectural changes to attenuate that issue, the authors in [11] proposed to create a data augmentation by mixing up images and then create two additional losses besides the normal classification loss so that the model can be trained to have localization capabilities and not only classification.

A more complex alternative to the CAM is the Grad-CAM proposed in [12]. The authors propose an approach to obtain the object localization using the gradients of the predicted class in the final convolutional layer and not only the layer information. The method can be applied on a wide variety of CNN without the need of performing any architectural changes or re-training. The main objective is the creation of an object detector on typical classification networks by producing visual explanations. However, they show that the method can be adapted to a wide range of applicabilities as textual image explanations, image captioning and visual question answering. For a sake of simplicity, in this thesis it will only be used the simpler CAM method.

2.2.2 Fire and smoke Detection

The need to create new and wiser methods of monitoring, detecting and fighting fire has led to the development of recent research. The detection part is the one that has powered the largest amount of approaches, through the use of vision-based systems. The different approaches can be separated in two ways, one based on the type of vision system used and the other on the type of techniques used [13].

Regarding the type of vision system, the approaches can be separated into terrestrial and aerial. The terrestrial systems use fixed-frame cameras [14] and present a major drawback which is the coverage they achieve. On the other hand, aerial systems using manned or unmanned aerial vehicles [15] [16] [17] allow a much greater coverage and can even cover inaccessible areas or areas considered too dangerous for the combat teams.

Regarding the type of techniques used, the approaches can be separated into classic and deep learning methods. The classic methods rely on typical computer vision techniques [18], where the detection is made based on the image characteristics, such as the colour patterns, texture and others. With the deep learning methods [19], the prediction is done using the features that ANN extract from several examples of similar images.

2.2.2.A Classic methods

The work done on fire and smoke detection based on computer vision presents a wide variety of methods. The big majority of them are based on colour, spatial and temporal features. These characteristics are very specific for fire compared to other objects. However, the same does not happen with smoke since it can get high similarities with other objects like clouds. Most of the approaches follow a common pipeline, first find moving pixels using background subtraction and then apply a colour model to find fire regions [20]. The base approach is to create a mathematical based model, defining a sub-space on a colour space that represents all the fire-coloured pixels in the image [21]. On this line, Wang et al. [22] proposed a method based on a Gaussian model learned in the YCbCr colour space. The major drawback of only colour based fire detection models is the high false alarm rates since single-colour information is insufficient in most cases so, on [23], the authors added texture analysis to create *Bowfire*.

By adding the temporal component to fire detection algorithms, video sequences [24] can be used instead of single frames. The authors in [25] added to the common baseline of finding the fire pixels with moving pixel detection and colour-space analysis, a wavelet analysis in the spatial and temporal domain. They make use of fire's frequency signature, with the idea that flames flicker with a characteristic frequency. So, they use the oscillation of the R component in the RGB image during a set of frames to create a wavelet. After filtering the wavelets and once all the components are fused, they can create a fire mask.

In a forest fire, the fixed cameras are placed on high altitude spots from which they can cover large

areas of forest terrain. One example operating in Portugal is *CICLOPE* proposed in [26]. The authors present a system of a tele-surveillance system that can perform remote monitoring and automatic fire and smoke detection. They use three algorithms for the detection. First, they use adaptive backgrounds for the background subtraction and detect potential targets. Then, they do another background subtraction but now comparing with past images to detect fire/smoke incidents. Finally, they do a statistical analysis of the RGB components in the image over time.

2.2.2.B Deep Learning methods

All of the previous methods depend heavily on the features delimited by the authors, which may make them too specific for a certain situation as concluded in [27]. Also, methods using motion tracking with background subtraction are limited to work properly only with fixed cameras. On the other hand, methods using deep learning methods, automatically learn which features are best for the given problem. This is why deep learning methods can outperform the classic methods. On [28] the authors do a comparative analysis between colour-model based methods versus deep learning methods. They use a logistic regression model, which is a very simple deep learning method and yet it is the one that obtains the best overall performance compared to all colour-based models. They also prove the robustness of these methods for colour changes and the presence of smoke. Thus, considering the superiority of the deep learning methods compared to those based on colour, several authors presented methods using CNNs to detect fire and/or smoke.

Still using surveillance cameras but now with deep learning methods, the authors in [29] present a solution for real-world surveillance scenarios using a computationally efficient CNN based fire detection system.

The authors from [30], use a CNN to first detect and localize, with a bounding box, regions in the image that can be fire candidates. Then, the regions are divided into patches that are classified by analyzing their spatial texture. This way, they can reduce the number of false positives on objects with similar colour patterns as fire.

In the most basic way, Q. Zhang et al. [31] propose a method where they do an image patch division and then classify them using a CNN. The final output is a set of patches in which fire is present which might not represent fire very well. The set of images of their dataset used for training does not even reach the two hundred images mark.

Q Zhang et al. [32] due to the lack of forest fire smoke images, created a dataset where they added two kinds of smoke, real smoke and simulative smoke, into forest background. Then they have used that to train a *Faster R-CNN* [33] that was later tested on real forest smoke images. The results seem to improve the detection performance even though the simulative smoke was not very realistic.

The lack of a good public accessible dataset for fire and smoke makes it hard to develop a good

deep learning technique. This problem is transversal and strongly highlighted by the vast majority of authors. The few existing datasets are highly biased towards the fire or smoke class and this does not portray the real situation wherein the vast majority of cases these events are not present. So, on [34] the authors created a small dataset highly unbalanced by including fewer fire images and more non-fire images. They have trained and tested two deep CNN, a VGG16 [35] and a ResNet50 [36] for classification both providing good performance.

The number of works carried out in the area of segmentation in fire and smoke situations are quite rare. This is due not only to the scarcity of datasets but also to the fact that these datasets are mostly annotated only at the image level. To develop a segmentation network, it is necessary to have annotations at the pixel level. If this type of annotation is already considerably expensive for normal objects, for the case of fire and smoke, which have a very irregular shape, it becomes even more expensive. That said, the number of weakly supervised segmentation approaches has been growing lately. In the big majority of these approaches, the authors make use of the information learned from the classification models to produce considerable accurate segmentation maps.

In [37] the authors create a CNN in which the final convolutional layer is a fuse of the 16 feature map from the previous layer into a single feature map. Then, by using a sliding window on that final feature map they can predict the presence of fire and smoke on each window and consequently create a detection mask in the whole image. The model has shown good classification performance, however, the detection masks are not precise and cannot correctly resemble the shape of fire and smoke.

In [38], the authors fuse the information from 3 selected feature maps of a convolutional layer, from a classification model, to create a mean activation map and then apply a binarization to it and create a segmentation mask. The resulting masks are not very precise, containing a considerable number of uncorrected classified pixels. In the end, they feed the image that was classified as fire to a *SqueezeNet* model [39] to predict the environment in which the fire is burning.

The scarcity of datasets in this area of work makes deep learning methods quite limiting. For the case of segmentation, this scarcity becomes even greater since the number of datasets with annotations at the pixel level is even smaller. The process of creating such labels is highly expensive. Therefore, in most works with this type of data, it is difficult to develop a robust method. Therefore, in this thesis, it is proposed to overcome this problem using weakly supervised supervised methods for fire and smoke segmentation. This way, one can use the few existing datasets and complete them with a large number of examples only annotated at the image level. This type of annotation has a minimal cost allowing to expand the datasets to acceptable levels for deep learning methods. However, all the works in this area using weakly supervised methods present a final detection with little precision and lack of detail. This work overcomes such limitation by combining weakly-supervised methods with post-processing, creating a segmentation mask with great detail that closely resembles the shape of fire and smoke.

3

Methodology

Contents

This chapter introduces and describes the proposal approach used to achieve the proposed objectives. It starts with an overall description of the system and how the information flows in the pipeline. Then, each method of the different system components is explained in a more detailed way.

3.1 Proposed Approach

The proposed approach intends to develop a system capable of detecting and localizing areas of fire and smoke in images using weakly supervised methods. The system is divided into two similar systems, one for fire and one for smoke. Both systems follow the same pipeline described in Figure 3.1 with the only difference being the parameters used in the different components.

The images will be taken from aerial vehicles equipped with visual-based equipment and will then be transmitted to the proposed system. Then, the image starts by being fed to a classification model the presence of fire/smoke in the whole image will be analyzed. If fire/smoke is detected then it goes on to the next phase, otherwise, the system ends here. The next phase consist in using the CAM method to extract the information that the classification model used to make the classification prediction and create a probabilistic heatmap. In this heatmap, areas with higher probabilities correspond regions where fire/smoke are likely to be present. Then, a binarization is applied to the heatmap to create a binary mask that can locate the fire/smoke location but cannot correctly represent it in terms of its shape. Therefore, a post-processing phase is then required. For that, the CRF method is used which takes as input the coarse and blob-like binary mask and the original image. By combining both inputs and analyzing the spatial and colour correlations, the method can create a segmentation mask at the pixel level that represents the location of the fire/smoke as well as its shape in a detailed way. In the following sections the methods used in each of the main blocks of proposed system will be describe in detail .

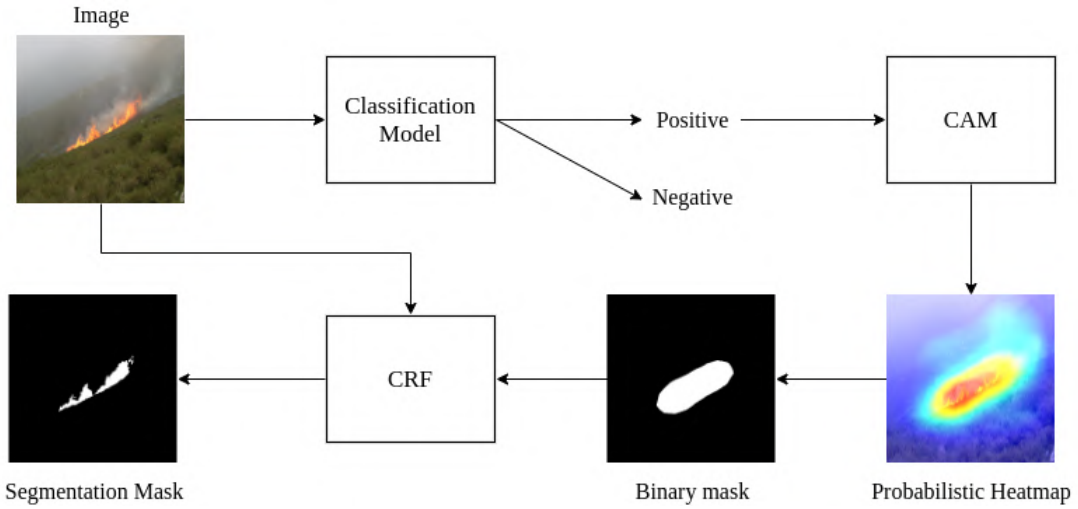


Figure 3.1: Overall proposed approach

3.2 Classification Model

For the classification model, the choice was to use the VGG network architecture proposed in [35] as a starting point. There are different VGG configurations according to the number of layers used, ranging from 11 weight layers to 19 weight layers. Figure 3.2 illustrates the largest architecture, the 19 weight layers version with the representative legend of the different layers.

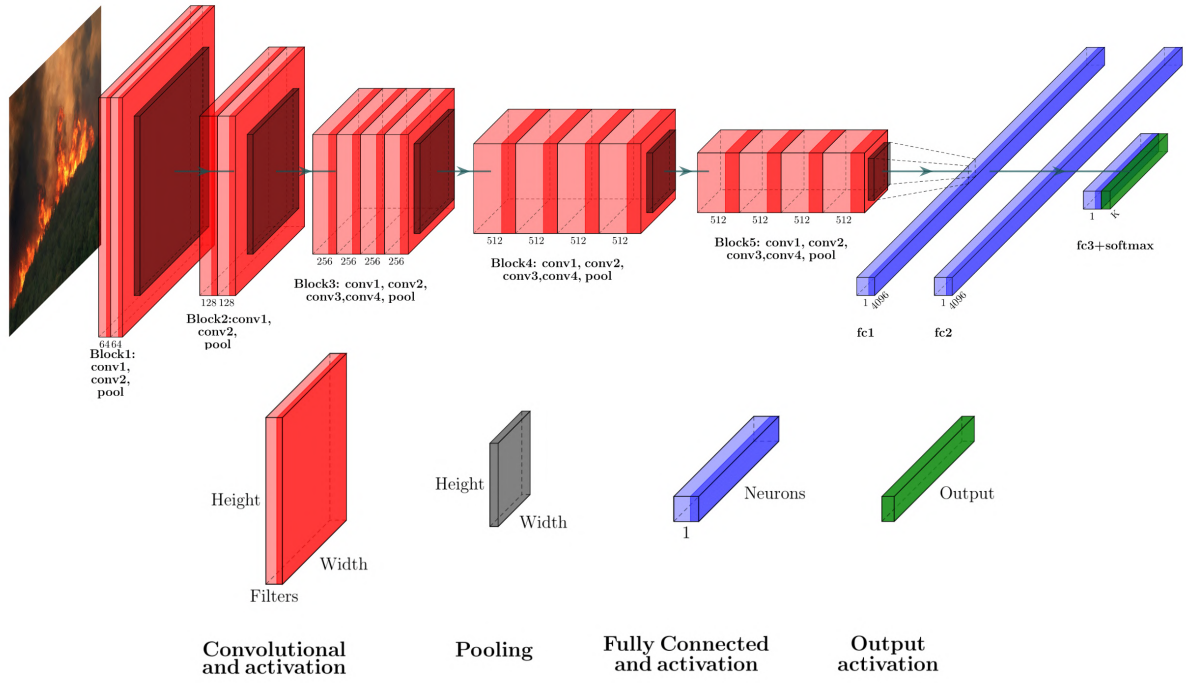


Figure 3.2: On top is the original VGG19 architecture and below is the corresponding legend for the different types of layers used

The network is composed of 5 main blocks of convolutional layers and then has 3 fully-connected layers. Each of the blocks contains a series of convolutional layers with kernels with a very small receptive field (3 $\times$ 3). The size of this kernel is the minimum size in which is possible to capture the notion of left/right, up/down, centre. The convolution stride is fixed to 1 pixel. In the end, each block has a max-pooling layer that downsamples the feature maps by summarizing the presence of features in patches. The pooling is performed over a 2 $\times$ 2 pixel window, with stride 2. This way on the next block, the size of the feature maps will be half of the previous one. Following the fifth block, the feature maps on the last convolutional layers are flattened to a vector with dimension one. The next two fully-connected layers contain the same size, corresponding to the number of pixels in the last max-pooling layer times the number of filters. All the previous layers are equipped with *ReLU* activation. The final layer is another fully-connected layer but with *Softmax* activation which is responsible to do the classification. The size of this layer is equal to the number of classes the model is designed to predict.

However, to apply the CAM algorithm, which will be explained in the next Section, it was necessary to perform some changes in the network. First, it is necessary to remove all fully-connected layers since they disrupt the spatial integrity maintained in the convolution layers. So, only the backbone is kept, all the layers after block5-pool were removed. Next, a global average pooling (GAP) layer is added to calculate the spatial average of each feature map in the last convolutional layer. In the end, one final fully connected layer with size two and *Sigmoid* activation function as in Equation (2.6) is added. Every image is first normalized to range in $[0 : 1]$ before entering the Input Layer. Figure 3.3 illustrates the adapted VGG19 model to the proposed approach.

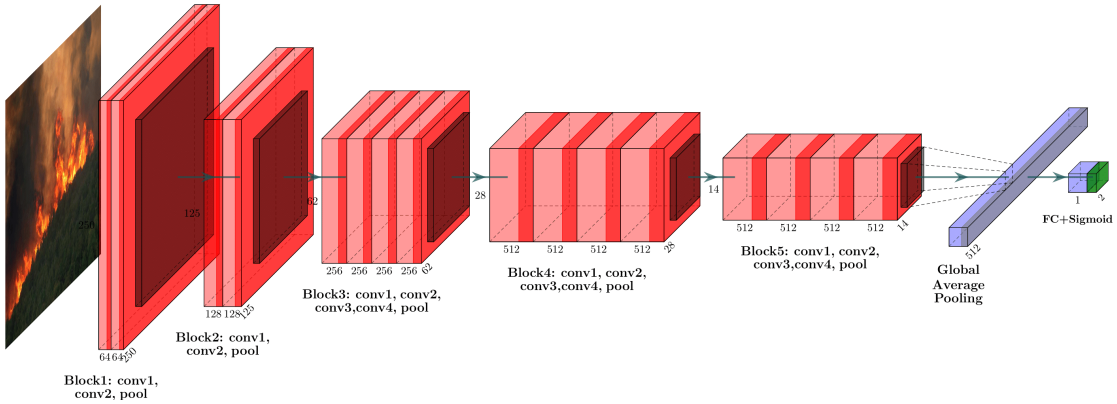


Figure 3.3: Adapted VGG19

For the fire model, the 19 weight layer version - VGG19 - was used while for the smoke model the 16-layer version - VGG16 - was used. In the end, for the VGG16 there is a mapping resolution of 16×16 and for the VGG19 a mapping resolution of 15×15. Table 3.1 and 3.2 describe the final architectures for the smoke and fire model respectively.

Table 3.1: Smoke model, using the backbone of VGG16

Layer		Type	Feature maps	Kernel size	Stride	Activation	Input	Output
Input		Image	3	-	-	-	-	256 x 256 x 3
block1	conv1	Convolutional	64	3 x 3	1	Relu	256 x 256 x 3	256 x 256 x 64
	conv2	Convolutional	64	3 x 3	1	Relu	256 x 256 x 64	256 x 256 x 64
	pool	Max Pooling	64	2 x 2	2	-	256 x 256 x 64	128 x 128 x 64
block2	conv1	Convolutional	128	3 x 3	1	Relu	128 x 128 x 64	128 x 128 x 128
	conv2	Convolutional	128	3 x 3	1	Relu	128 x 128 x 128	128 x 128 x 128
	pool	Max Pooling	128	2 x 2	2	-	128 x 128 x 128	64 x 64 x 128
block3	conv1	Convolutional	256	3 x 3	1	Relu	64 x 64 x 128	64 x 64 x 256
	conv2	Convolutional	256	3 x 3	1	Relu	64 x 64 x 256	64 x 64 x 256
	conv3	Convolutional	256	3 x 3	1	Relu	64 x 64 x 256	64 x 64 x 256
	pool	Max Pooling	256	2 x 2	2	-	64 x 64 x 256	32 x 32 x 256
block4	conv1	Convolutional	512	3 x 3	1	Relu	32 x 32 x 256	32 x 32 x 512
	conv2	Convolutional	512	3 x 3	1	Relu	32 x 32 x 512	32 x 32 x 512
	conv3	Convolutional	512	3 x 3	1	Relu	32 x 32 x 512	32 x 32 x 512
	pool	Max Pooling	512	2 x 2	2	-	32 x 32 x 512	16 x 16 x 512
block5	conv1	Convolutional	512	3 x 3	1	Relu	16 x 16 x 512	16 x 16 x 512
	conv2	Convolutional	512	3 x 3	1	Relu	16 x 16 x 512	16 x 16 x 512
	conv3	Convolutional	512	3 x 3	1	Relu	16 x 16 x 512	16 x 16 x 512
	pool	Max Pooling	512	2 x 2	2	-	16 x 16 x 512	8 x 8 x 512
GAP		Global Average Pooling	512	-	-		8 x 8 x 512	1 x 1 x 512
Output		Fully connected	2	-	-	Sigmoid	1 x 1 x 512	2

Table 3.2: Fire model, using the backbone of VGG19

Layer		Type	Feature maps	Kernel size	Stride	Activation	Input	Output
Input		Image	3	-	-	-	-	250 x 250 x 3
block1	conv1	Convolutional	64	3 x 3	1	Relu	250 x 250 x 3	250 x 250 x 64
	conv2	Convolutional	64	3 x 3	1	Relu	250 x 250 x 64	250 x 250 x 64
	pool	Max Pooling	64	2 x 2	2	-	250 x 250 x 64	125 x 125 x 64
block2	conv1	Convolutional	128	3 x 3	1	Relu	125 x 125 x 64	125 x 125 x 128
	conv2	Convolutional	128	3 x 3	1	Relu	125 x 125 x 128	125 x 125 x 128
	pool	Max Pooling	128	2 x 2	2	-	125 x 125 x 128	62 x 62 x 128
block3	conv1	Convolutional	256	3 x 3	1	Relu	62 x 62 x 128	62 x 62 x 256
	conv2	Convolutional	256	3 x 3	1	Relu	62 x 62 x 256	62 x 62 x 256
	conv3	Convolutional	256	3 x 3	1	Relu	62 x 62 x 256	62 x 62 x 256
	conv4	Convolutional	256	3 x 3	1	Relu	62 x 62 x 256	62 x 62 x 256
	pool	Max Pooling	256	2 x 2	2	-	62 x 62 x 256	31 x 31 x 256
block4	conv1	Convolutional	512	3 x 3	1	Relu	31 x 31 x 256	31 x 31 x 512
	conv2	Convolutional	512	3 x 3	1	Relu	31 x 31 x 512	31 x 31 x 512
	conv3	Convolutional	512	3 x 3	1	Relu	31 x 31 x 512	31 x 31 x 512
	conv4	Convolutional	512	3 x 3	1	Relu	31 x 31 x 512	31 x 31 x 512
	pool	Max Pooling	512	2 x 2	2	-	31 x 31 x 512	15 x 15 x 512
block5	conv1	Convolutional	512	3 x 3	1	Relu	15 x 15 x 512	15 x 15 x 512
	conv2	Convolutional	512	3 x 3	1	Relu	15 x 15 x 512	15 x 15 x 512
	conv3	Convolutional	512	3 x 3	1	Relu	15 x 15 x 512	15 x 15 x 512
	conv4	Convolutional	512	3 x 3	1	Relu	15 x 15 x 512	15 x 15 x 512
	pool	Max Pooling	512	2 x 2	2	-	15 x 15 x 512	7 x 7 x 512
GAP		Global Average Pooling	512	-	-		7 x 7 x 512	1 x 1 x 512
Output		Fully connected	2	-	-	Sigmoid	1 x 1 x 512	2

3.3 Class Activation Mapping

Image Segmentation is a demanding process that involves assigning a class to every pixel in an image. The typical methods of segmentation rely on a large set of images and the corresponding fully-supervised ground-truth (GT) masks. The creation of a fully-supervised dataset is time-consuming and can be quite ambiguous. Detailed image annotations such as bounding boxes or pixel-level masks are both subjective and very expensive to gather or create.

For these same reasons it was decided to use weakly supervised methods, in which there is only the need for image-level labels indicating if the class is present somewhere in the whole image. More specifically, the CAM algorithm proposed by [1] was used. It is based on the idea that a classification CNN develops localization capabilities, despite being trained only with image-level labels.

So, taking an off the shelf classification CNN, it is necessary to understand which are the features of the image that the model uses to classify it into the predicted class. Then, translating those features into zones in the image, one can produce an estimate of the localization prediction.

For that, only the backbone of the CNN will use and the classifier with fully-connected layerswill be discarded since they remove the spatial integrity maintained in the convolution one. Next, a GAP layer is added to calculate each feature map's spatial average in the last convolutional layer. The use of a GAP like in [1] instead of a global max pooling, as proposed in [40], is because the first one encourages the network to identify the whole extent of the object while the latter one only encourages the network to identify only the most discriminant part. In the end, a final layer is added for the prediction.

The idea of adding a GAP layer after the convolutional layers is to summarize each feature map in the last layer into a node. So, each node represents a feature map that represents a region in the image. To perform the classification, each node will be weighted according to the relevance of the feature map it represents for the prediction. A feature map can be weighted positively if the visual pattern that it represents is relevant for the output or negatively if not. So, one can make a weighted sum of each feature from the last convolutional layer to produce a heatmap as:

$$H_c(x, y) = \sum_{i=1}^{n} w_i^c f_i(x, y). \tag{3.1}$$

Where H represents the CAM heatmap with the predicted location and w_i^c is the weight of the activation of the i^{th} feature map $f(x, y)$ for the predicted class c.

The heatmap will highlight the zones in the image that the model has used for the prediction and thereby those are the zones where the class is more probable to be present. Figure 3.4 illustrated the overall method. For that example, the model has predicted the presence of the class "Australian terrier" in the image and then with the use of the features from the last "conv" layer it is possible to get a heatmap highlighting where the class "Australian terrier" is more probable to be present in the image.

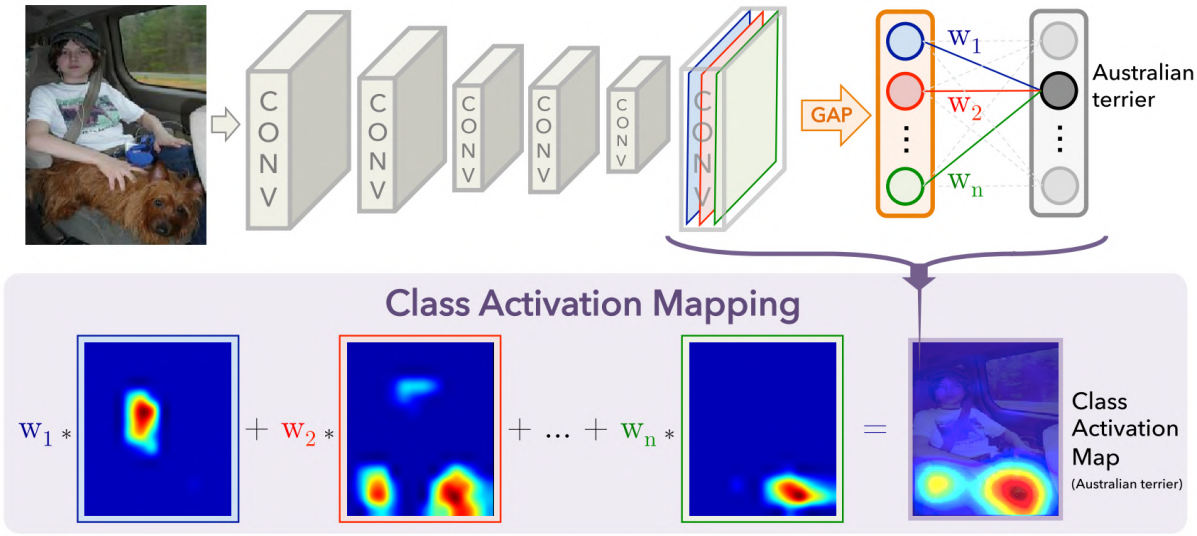


Figure 3.4: CAM approach from [1]

3.4 Conditional Random Fields

Conditional Random Fields (CRF) are a class of discriminative models that are used for prediction using information from the neighbouring samples. It models the data into a graphical model and does the prediction through an iterative energy minimization process.

CRF can be used for segmentation tasks [41] [42] either by itself or in combination with other segmentation techniques for example with deep learning neural networks [43] [2]. When by itself the CRF uses traditional hand-crafted features as a prior and when conjugated with other techniques it relies on them to provide the features and then act as post-processing. In the latter, the goal is for CRF to help creating a more detailed segmentation as illustrated in Figure 3.6.

In a post-processing situation, the image can be seen as a graph, as in Figure 3.5, where each pixel is perceived as a node and the nodes are connected with edges ξ. Each node can have a finite set of states corresponding to the possible classes and each state has a unary cost $\psi_u(x_i)$ for each pixel. The pairwise cost $\psi_p(x_i, x_j)$ between nodes is determined by the spatial and color distance within the two pixels i and j. The graph may be built as a grid where only adjacent pixels are connected to each other, or fully-connected, as in this case, where each pixel is connected to all other pixels in the image. Finally, the assignment of each pixel to label is treated as an energy minimization problem where the energy corresponds to the sum of the total unary and pairwise costs as in Equation (3.2). It is an iterative process wherein at each inference step the energy is progressively minimized.

The Fully Connected Conditional Random Fields (FCCRF) uses neighbouring context to predict the class of a pixel. In a fully connected situation every pixel in an image can be used to determine the class of one pixel. The energy function can be described as:

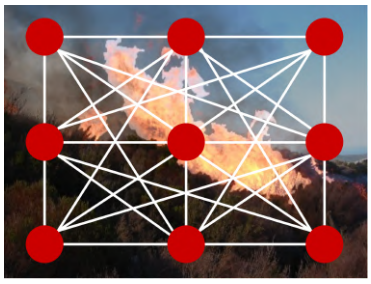


Figure 3.5: CRF graphical model

$$E(x) = \sum_i \psi_u(x_i) + \sum_{i,j} \psi_p(x_i, x_j), \tag{3.2}$$

where x represents the set of labels corresponding to each pixel, i and j range from 1 to N being N the size of the image, $\psi_u(x_i)$ is the unary potential and $\psi_p(x_i, x_j)$ is the pairwise potential.

The unary potential $\psi_u(x_i)$ sets a cost of assigning a label x_i to pixel i with a probability $P(x_i)$. This probability is provided by the classification model. The probability decides how much weight the unary mask should have in the energy function $E(x)$. The potential is then described as:

$$\psi_u(x_i) = -log(P(x_i)), \tag{3.3}$$

where $P(x_i)$ is the pixel probability at pixel i.

The pairwise potential $\psi_p(x_i, x_j)$ sets a cost to assign the label to pixel i in pairs, i.e. the cost of pixel i will be according to pixel j. It will analyze the neighbouring pixels to predict the class for pixel i. This potential has the form of a linear combination of Gaussian kernels as:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \underbrace{\sum_{m=1}^{K} w^{(m)} k^{(m)}(f_i, f_j)}_{k(f_i, f_j)}, \tag{3.4}$$

where the term $\mu(x_i, x_j)$ is a label compatibility function which is responsible for introducing a penalty for nearby pixels that are assigned different labels: $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$. Each $k^{(m)}(f_i, f_j)$ is a Gaussian kernel between the feature vectors f_i, for pixel i, and f_j for pixel j. The $w^{(m)}$ acts as a weight factor defining the importance of each Gaussian in the linear combination.

For image segmentation the $k(f_i, f_j)$ is a contrast-sensitive two-kernel potentials as:

$$k(f_i, f_j) = w^{(1)} \underbrace{exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right)}_{appearance\ kernel} + w^{(2)} \underbrace{exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right)}_{smoothness\ kernel}. \tag{3.5}$$

It consists of a weighted sum of a position and colour sensitive double Gaussian with weight $w^{(1)}$ with

a position-sensitive single Gaussian with weight $w^{(2)}$. The first Gaussian is the appearance kernel and it controls the degrees of nearness and similarity with the idea that nearby pixels with similar colours are likely to belong to the same class. The first term depends on both pixel positions p_i and p_j and the scale of the Gaussian is controlled by the spatial standard deviation θ_α. The larger the standard deviation, the flatter the Gaussian and furthest pixels will be taken into account. The second term depends on both pixel colour intensities I_i and I_j and the scale of the Gaussian is controlled by the colour standard deviation θ_β. As before, the larger the standard deviation, the flatter the Gaussian and wider is the range of colours that will be taken into account. The second Gaussian is the smoothness kernel and it removes small isolated areas giving a sharper boundary delimitation. It only depends on both pixel positions and is controlled by the smoothness standard deviation θ_γ. This standard deviation is a similar behaviour as the first one.

Figure 3.6 illustrates the overall process of the CRF algorithm as a post-processing technique. As inputs, it takes a segmentation mask that will be used for the unary potential and the Original image. Then by conjugating both, it sets the energy function. After minimizing the function, the output is a mask with plenty of detail, thus showing how CRF can improve segmentation.

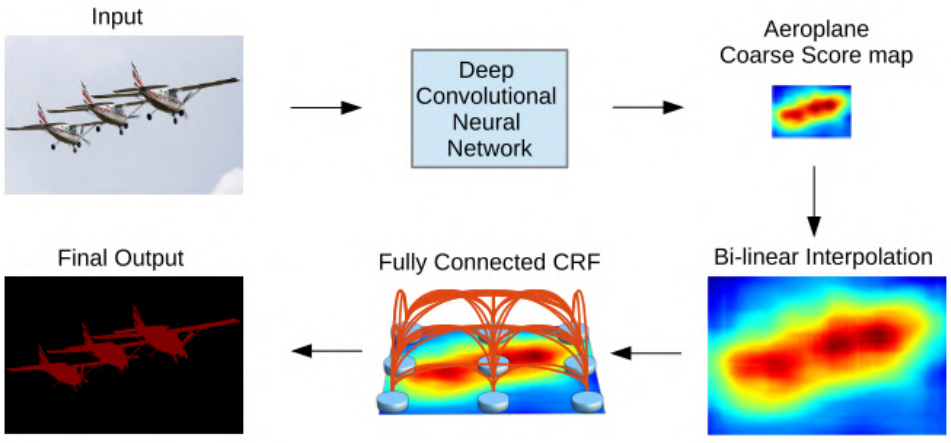


Figure 3.6: Example of the use of the CRF as a post processing technique from [2]

4

Experiments

Contents

This chapter starts by introducing the weakly-supervised dataset used for model development as well as the description of a small fully-supervised dataset used for the segmentation tests. It is also introduced the metrics used to evaluate the classification and segmentation. Furthermore, it analyzes and states the setup procedures taken to develop and optimize the methods that compose the proposed system. Additionally, it is described the work environment in which the system was develop regarding both hardware and software. In the end, it is listed all the experimental results that will be done to validate and evaluate the overall performance of the system.

4.1 Dataset

The first and most important step for computer vision and deep learning approaches is the creation of a complex and diverse dataset of images. This way and applied to the wildfire situation, it is possible to create a robust method capable of classifying and subsequently segmenting various forest fire scenarios. Different models will be trained, one for fire detection and the other for smoke detection. Thus, it is necessary to have a dataset that can cover both situations.

Two datasets were created, one with annotations at the image-level to train, validate and test the classification model and another with annotations at the pixel-level to validate and test the segmentation approach.

4.1.1 Image-level dataset

Given that the number of publicly available forest fires datasets is quite scarce, one had to be created. As a starting point for the fire examples, the dataset [44] was used considering that it contains good examples of forest fires as well as controlled fires. Since this dataset only contained positive images of fire, it was necessary to augment it with negative examples such as samples of forest images without fire as well as strong negatives like sunsets. All the negative examples were gathered manually from the web. For the smoke examples, the data from [45] and from [46] was used as starting point and it was then augment it with individual images gathered from the web. As described in Table 4.1, in total, only 40% of dataset images were from free available datasets and the remaining 60% were gathered by hand which shows the scarcity of good and freely available datasets

Table 4.1: Dataset images source description

Dataset source	Percentage [%]
Available online	40
Manually gathered	60

The models are exclusively trained for classification purposes at the image-level so the dataset used for training is exclusively labelled at this level. With image-level labels is important to have examples

with a single class present so that the net can learn by itself what is fire and what is smoke independently. The models behave as a one-vs-all classifier and so there will be an individual label for each class. In neither case is there any indication of its location. Therefore, an image can be labelled independently as containing fire and smoke. Figure 4.1 shows some examples of labelled images. Figure 4.1(a) does not contain neither fire nor smoke so is labelled as negative for fire and negative for smoke. Figure 4.1(b) contains only fire and so is labelled as positive for fire and negative for smoke, and the same in Figure 4.1(c) but for smoke. Finally, 4.1(d) contains both fire and smoke and so is labelled as positive for both labels.

(a) Fire: Negative, Smoke: Negative

(b) Fire: Positive, Smoke: Negative

(c) Fire: Negative, Smoke: Positive

(d) Fire: Positive, Smoke: Positive

Figure 4.1: Dataset examples and the corresponding labels

All the images in the dataset were labelled at the image-level by hand by the author of this thesis. The image gathering and the consecutive labelling process is almost effortless and very fast. Collecting annotations at the image level has a much lower cost than trying to do it at the pixel level, in which one has to highlight the regions of interest by hand.

Table 4.2 presents an overview of the created dataset. The labelling method used, made it possible to divide the whole dataset into two classes separately. In this way, the presence of each class in the image can be assessed independently.

Table 4.3 summarizes the split of the dataset for the classification model training, validation and testing. The class percentages are also shown for each one of them.

Table 4.2: Dataset composition

# Images	Label		Percentage [%]
1807	Fire	Positive	70
		Negative	30
	Smoke	Positive	70
		Negative	30

Table 4.3: Dataset split percentages

Set	Split [%]	Fire		Smoke	
		Positive [%]	Negative [%]	Positive [%]	Negative [%]
Train	78	69	31	71	29
Val	12	70	30	69	31
Test	10	75	25	65	35

4.1.2 Pixel-level dataset

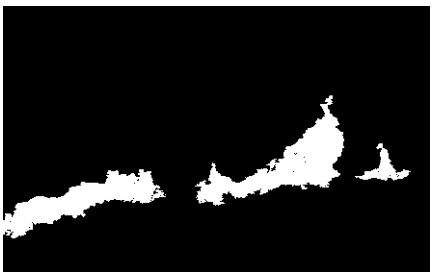
In addition to previous dataset, there was the need to create a smaller dataset with labels at the pixel level to validate and test the weakly supervised segmentation approach. This dataset was never used to train any model.

This set is composed of images from both classes and their ground truth, as in Figure 4.2, and its composition is described in Table 4.4. For the fire examples, the starting point was once again the dataset of [44] since it also contains the ground truth masks of the images. Regarding the smoke examples, the starting point was [45], to which there had to be some post-processing to get simple binary masks. Both datasets were augmented with an internal dataset created by the team of the Firefront project. It should be noted that the number of images annotated at the pixel level is approximately half of the images of the previous dataset. Furthermore, the number of examples of this set would be significantly smaller without the segmentations performed by the Firefront team. This emphasizes the limited amount of quality datasets in this area of work, which becomes even scarcer with pixel-level annotations.

Table 4.4: Annotated dataset description

Class	# Images and GT
Fire	600
Smoke	260

(a) Fire image



(b) Fire image ground truth

(c) Smoke image

(d) Smoke image ground truth

Figure 4.2: Dataset examples annotated at the pixel level

4.2 Metrics

The performance of a classification model is analysed by comparing the predicted values and the actual values in a test set. In binary classification, this performance can be summarized by using a confusion matrix as the one in Table 4.5. In a confusion matrix, there are four possible events.

Table 4.5: Confusion matrix

		Actual	
		Positive	**Negative**
Predicted	**Positive**	true positive (TP)	false positive (FP)
	Negative	false negative (FN)	true negative (TN)

A detection is considered to be TP if the prediction is positive and it is equal to the ground truth or FP if it is also positive but differs from the ground truth. The same happens for negative examples, where a TN is when the prediction and the ground truth matches being both negatives and a FN when they differ, the prediction is negative but the ground truth not.

With all the events defined, one can estimate different metrics to evaluate the model:

- **Accuracy** - it is the ratio between the number of correct predictions and the total number of

samples. It simply measures how often the classification model makes the correct prediction.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

- **Precision** - it represents the ratio between correctly predicted positives and the total number of predicted positives. It quantifies from all the positive predictions, how many are actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

- **Recall** - it represents the ratio between correctly predicted positives and the total number of ground truth positives. It shows the ability of the model to find all relevant instances in the test set, giving the percentage of images that were predicted correctly.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

The performance for a segmentation approach consists of comparing the ground truth mask with the predicted mask at the pixel-level. A mask contains a label for each pixel separating the foreground class from the background. The metrics to be used are:

- **intersection over union (IoU)** - also known as Jaccard Index and it is the intersection area divided by the union area of the predicted and ground truth masks.

$$IoU = \frac{area\,Overlap}{area\,Union} \tag{4.4}$$

- **mean intersection over union (mIoU)** - is the mean of the IoU for the foreground and background classes.

Additionally to the aforementioned classic metrics, to evaluate the results in a different perspective, one can analyse their distributions using a box plot as:

- **Box Plot** - useful to illustrate data distribution. It gives information on the variability or dispersion of the data, giving a good indication of how the results are spread out. The interquartile range (IQR) is defined as the distance between the first quartile $Q1$ (25^{th} percentile) and the third quartile $Q3$ (75^{th} percentile). Inside, there is the median of the data distribution. The top and bottom whiskers are defined as functions of the first and third quartile. The bottom one is equal to $Q1 - 1.5 * \text{IQR}$ and the top one is equal to $Q3 + 1.5 * \text{IQR}$. Any value outside of these whiskers is considered to be an outlier.

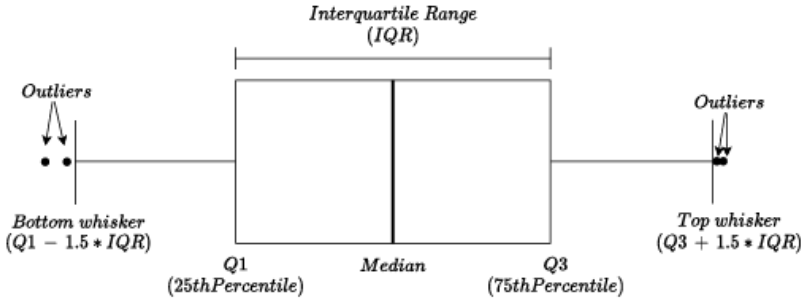


Figure 4.3: Box plot

4.3 Model Development

Two models were used, one to detect fire and one to detect smoke. The models were trained only for classification but were used later for weakly supervised segmentation. In this way, it is necessary that when training and validating the model its results in terms of localization are also monitored.

Therefore, during training and validation, several hyperparameters were tested until the best configurations were found. The chosen models represent the best classification and localization results. In the following subsections we provide a detailed description of the training strategies adopted to develop the various components of our model.

4.3.1 Classification Stage

The classification stage is a crucial part of the proposed approach, since it is the entry point of the proposed approach and it will decide which image should be analyzed for segmentation. So, the image will only move forward to the weakly supervised segmentation and then to the post-processing stage if it is classified as positive. Any image that is classified as negative will be discarded. Therefore, it is necessary to have a good classification phase to prevent any FP or FN. For example, one wants to avoid situations where the image contains fire but the model classifies it as negative and consequently, the fire would not be detected through the following stages. Also, one does not want to have a FN where there will be the risk of segmenting non-fire or non-smoke zones as being fire or smoke.

Both fire and smoke models have an output similar to the labels in the dataset which means that for both models the output of the network is fire or not fire and smoke or not smoke. It is important to have this two classes into consideration for the classification so that the model can learn to differentiate both classes during training and do not correlate them as a single class since they occur together very frequently. However, it was noticed that small changes in the training phase would result in different localization performances with the CAM method. So, the best performance is achieved with two different models and parameters. For the fire model it is only taken into account the fire output class and identical for the smoke model.

The classification phase proved to be a great challenge due to the labelling method used. The images are labelled at the image-level and not at the pixel level neither using bounding boxes so it becomes harder for the network to collect class-specific features. When training the network with this dataset using labels at the image level, each image is telling the network that the class for which it is labelled is present in the entire image. Thus, the network will learn features of the entire image itself, and not just the specific class as one would expect in a typical classification situation. Considering that we are working on an aerial image setting, there will always be much more information in a single image than just the class itself. Especially, in aerial images of fire and smoke, there will be common co-occurring objects such as vegetation, clouds, etc. To meet this challenge, it is necessary to collect a very diverse and complex dataset. Only this way it is possible for the network to distinguish class-specific features of fire and smoke that are common alongside the images in the dataset.

For both models, the pre-trained weights on the ImageNet [47] were used as a base point. This way the network training time is reduced and the training performance is typically better.

Table 4.6 shows the model parameters that lead to the best results, taking into account the classification results and visual perception of the localization results after applying the CAM method.

Table 4.6: Models parameters

Parameter	**Fire Model**	**Smoke Model**
Base Model	VGG19	VGG16
Optimizer	Adam	Adam
Learning Rate	1e-05	1e-06
Loss	Binary Crossentropy	Binary Crossentropy
Batch size	32	32
Early Stopping	patience = 10	patience = 10
Monitor	Validation Loss	Validation Loss

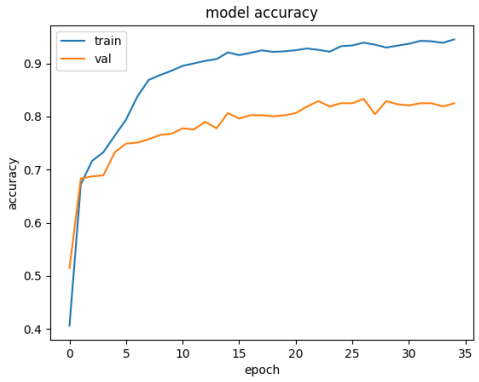
The choice of optimiser will have an impact not only in the classification but also in the localisation results. The optimizer, during training, minimizes the cost function by successively changing the weights of the network and each optimizer does this change in a different way, as explained in Chapter 2, resulting in different final weights. Thus, one must choose an optimizer that can not only correctly minimize the cost function but also do a good job of assigning weights to feature-maps that contain features specific to fire/smoke and not to other surrounding objects. This correct assignment will be critical for the CAM algorithm. The one that produced better results on both models was the *Adam* optimizer with a learning rate of $1e^{-5}$ and $1e^{-6}$ for the fire and smoke model respectively.

The loss function used only takes into account the performance in terms of classification at the image-level. Considering that the dataset used has only two classes fire or smoke, a *Binary Cross Entropy* was used and the output of the models indicates the image probability to contain fire and/or smoke. So, the problem is separated into two binary classification problems, one for each class. The loss function is

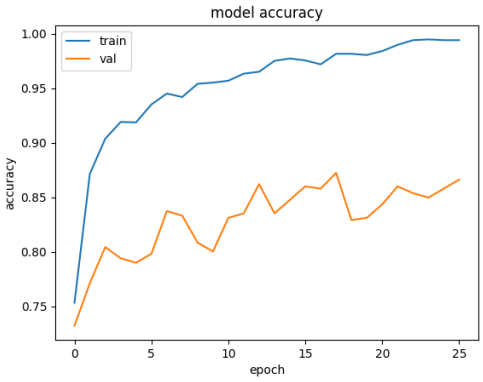
therefore the average of two binary losses.

For both models, early stopping was used with a patience of 10 and the monitoring factor was the validation loss. That is, during the training, if the validation loss does not decrease during 10 epochs, the training is stopped and the weights are saved. This way it avoids overfitting the training data.

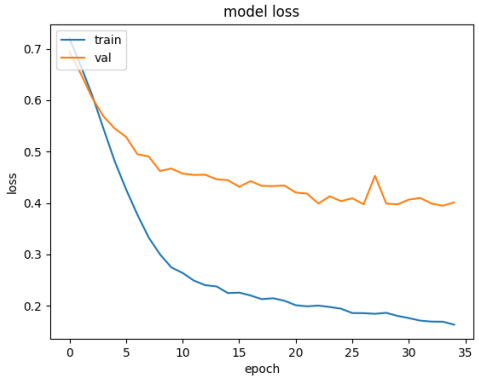
Figure 4.4 shows the progress during the training process in terms of accuracy and loss for the training and validation set. Both graphs show a good evolution in terms of accuracy and loss since the accuracy increases while the loss decreases. The smoke model achieved higher values in terms of accuracy and lower loss values than the fire model. However, the gap between the training and validation set is a little bigger. It is also notorious that, especially for the fire model, the accuracy values are not as high as expected in a normal classification situation, however, this can be explained by the fact that the model is performing classification in the whole image and thus, the number of class-specific features might not as high as normal. Also, the lack of fully connected layers makes the classification a harder challenge.



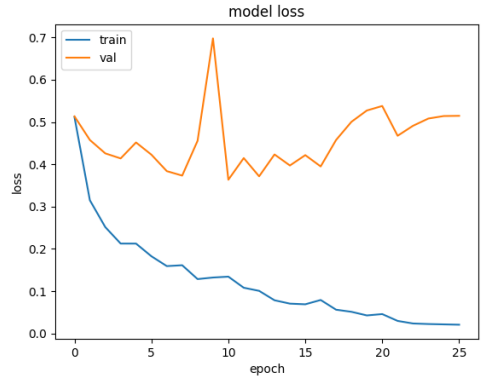
(a) Fire model accuracy



(b) Smoke model accuracy



(c) Fire Model loss



(d) Smoke Model loss

Figure 4.4: Models training progress with accuracy and loss

The class prediction is done at the image-level so, the result of passing an image to the models is a simple probability if there is fire/smoke in the whole image. To this probability, it is necessary to apply a classification threshold to consider the result as positive or negative. This way any prediction with a probability above the threshold τ is considered positive and in the same way, if it is below it is considered negative:

$$p(I) = \begin{cases} 0, & if\ p(I) < \tau \\ 1, & if\ p(I) \geqslant \tau \end{cases}, \tag{4.5}$$

where I is the image given to the model, $p(I)$ is the resulting model output probability of image I and τ is the threshold value. For both models, τ was set to 0.7. In Experiment B in Section 5.2 it is done a more detailed experiment to evaluate both classification models including the classification threshold.

Figure 4.5 and 4.6 illustrates six examples of predictions for the fire and smoke model at the image level. On 4.5(a), 4.5(b), 4.6(a) and 4.6(b) are demonstrated four TP examples where the models correctly predict the presence of fire/smoke. On 4.5(c) and 4.6(c) are FP examples where the models predict positive for fire and smoke while the ground truth is negative. These examples illustrate one issue with the image-level labels for the fire and smoke scenario. Fire and smoke are phenomenons that occur in their great majority coupled with each other, so it is a major challenge for the model to distinguish them when using image-level labels. On 4.5(d) and 4.5(e) are two TN examples that are very important to correctly classify since they co-occur very frequently in a fire situation, forest and firefighting airplanes. On 4.5(f) is also an TN example, illustrated with a sunset which contains a very similar colour pattern as fire. Similarly for the smoke situation, 4.6(d) and 4.6(e) represent TN examples on common co-occurring objects, forests and clouds. 4.6(f) can be considered as a FN, even though a very demanding one, where there is a small presence of smoke but the models predicted it as negative. It is a good example to show that the model can still distinguish smoke from fire since in this case the image clearly contains fire.

(a) Probability = 1.0 **(b)** Probability = 1.0 **(c)** Probability = 0.77

(d) Probability = 0.0 **(e)** Probability = 0.0 **(f)** Probability = 0.0

Figure 4.5: Examples of images classification and their respective probability of containing fire according to the Fire Model: (a) and (b) are TP examples, (c) is a FP example, (d) and (e) are TN examples and (f) is a FN example.

(a) Probability = 1.0 **(b)** Probability = 1.0 **(c)** Probability = 0.6

(d) Probability = 0.0 **(e)** Probability = 0.0 **(f)** Probability = 0.0

Figure 4.6: Examples of images classification and their respective probability of containing smoke according to the Smoke Model: (a) and (b) are TP examples, (c) is a FP example, (d) and (e) are TN examples and (f) is a FN example.

4.3.2 Weakly Supervised Segmentation stage

To extract the location of fire and smoke from the Classification model described in Section 4.3.1, the CAM algorithm explained in Chapter 3 was used. By summing representative features and subtracting unrepresentative ones, it is possible to highlight the image regions that the network used to predict the class. Consequently, these are the image regions where fire/smoke is more probable to be located. Figure 4.7 illustrates the overall CAM process for the fire model. In the end, there is illustrated the final heatmap for the input image.

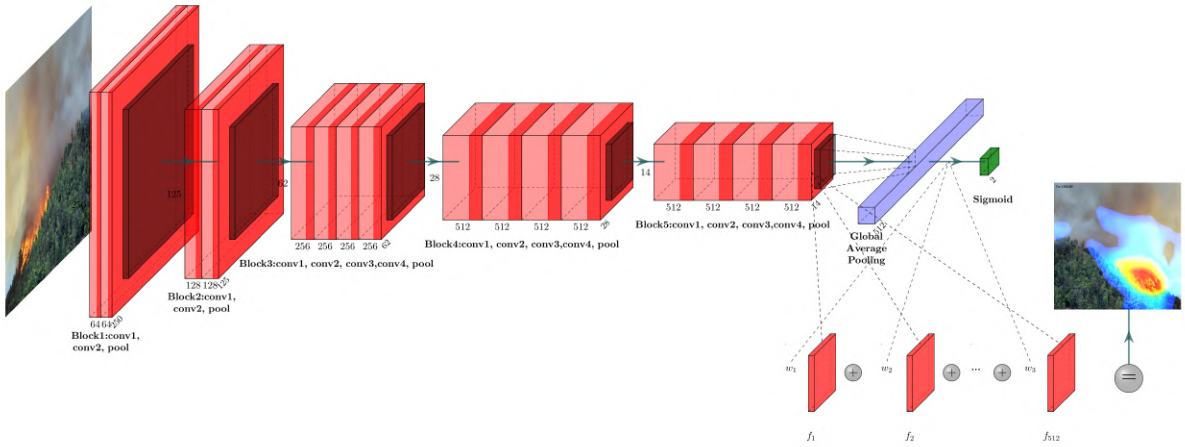


Figure 4.7: Overall CAM approach for the fire model

The probability heatmap is obtained by doing a weighted sum of the feature maps from the last convolutional layer (block5-conv4 for the fire model and block5-conv3 for the smoke model) as in Equation (3.1) and is illustrated for the fire situation in Figure 4.8. The *jet* colourmap, as in Figure 4.9, was chosen to illustrate the probabilities, where a red colour corresponds to a high probability and the blue to a low probability. So, in the heatmap, a high probability represents an area in the image very probable to contain fire/smoke

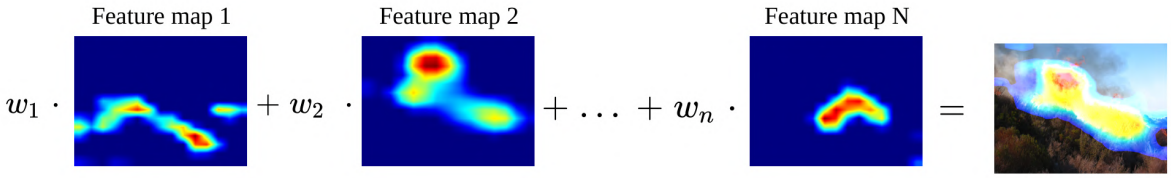


Figure 4.8: CAM weighted feature maps sum

For the fire case, and following what was proposed in [48], all feature maps with negative associated weight were discarded since these were being associated with fire zones rather than background. In Experiment C in Section 5.3 a more detailed analysis is done about the influence of feature maps with

Figure 4.9: Jet colourmap

negative weight to obtain the CAM mask. So, there will be no subtraction of feature maps, only those with positive weights were used for Equation (3.1), resulting:

$$H_c(x,y) = \sum_{i=1}^{n} w_i^c f_i(x,y), \quad if\ w_i^c > 0 \tag{4.6}$$

For smoke, the subtraction of feature maps with negative weights is beneficial.

The heatmap is created with the same mapping resolution of the last convolutional layer and is then upsampled to the original image size using bilinear interpolation. The final heatmap behaves as an object detector despite no supervision on the location was provided. Figure 4.10 shows some examples of the CAM heatmap for the fire model and Figure 4.11 for the smoke model. In both figures, it can be seen that the heatmaps assign a high probability in the correct location of fire and smoke and their respective extent despite the model has never been trained for that task.

The following step consists on transforming the heatmaps into a binary mask. So, the probabilistic heatmap was thresholded according to its maximum values as:

$$\theta = \alpha \max(H), \tag{4.7}$$

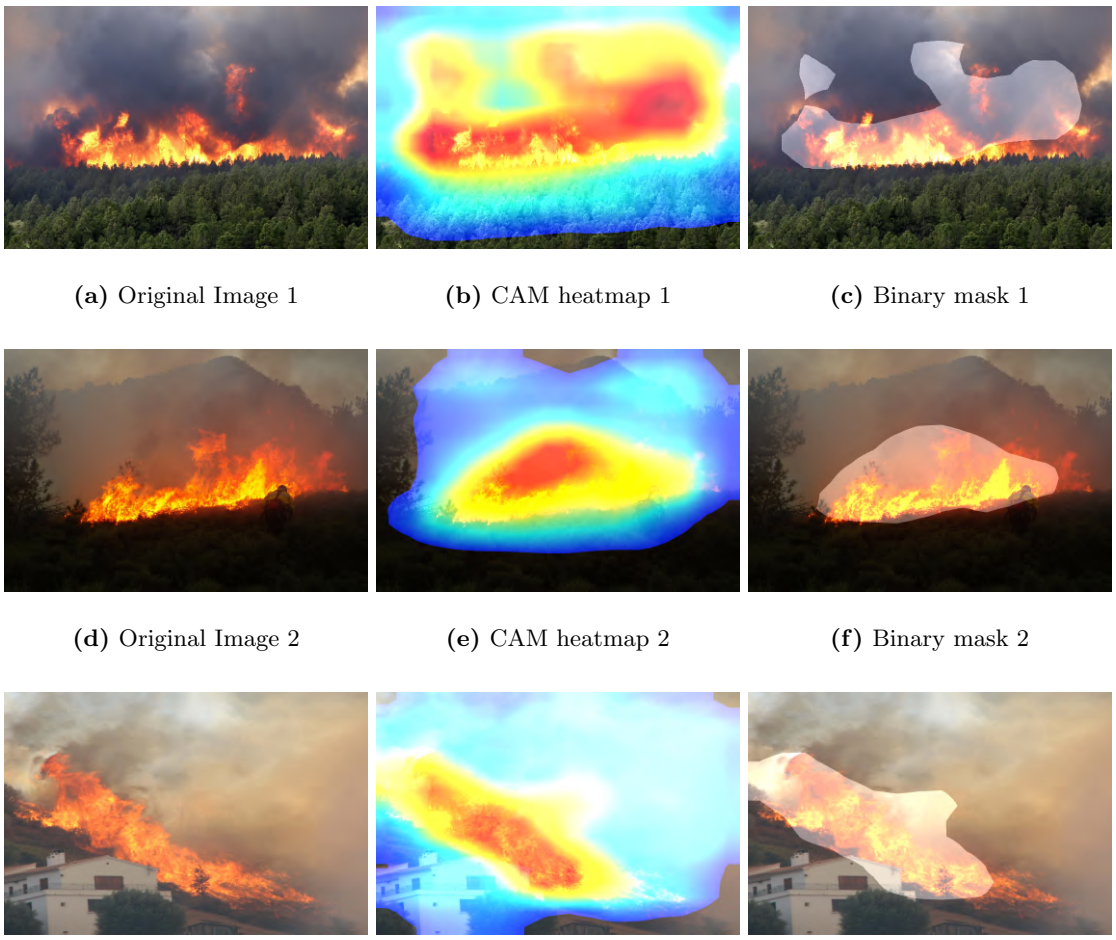
where θ is the thresholding value, α is a real between 0 and 1 and H is the probabilistic heatmap. Every pixel in the heatmap that has a probability superior to the threshold was set to 1 and below set to 0, as:

$$M(x,y) = \begin{cases} 1 & if\ \ H(x,y) \geq \theta \\ 0 & if\ \ H(x,y) < \theta \text{'} \end{cases} \tag{4.8}$$

where $H(x,y)$ is the pixel probability value on the heatmap H at position (x,y) and M is the resulting segmentation mask after thresholding.

For the fire model, the α was set to 0.5 while for the smoke one was set to 0.2. In Experiment D in Section 5.4 it will be done a comparison of different types of thresholding and their respective results. Also in Figure 4.10 and 4.11, there are represented the resulting segmentation mask after applying the threshold method. The brighter areas illustrate a positive label for fire/smoke.

Nevertheless, these heatmaps lack detail, representing rather rounded shapes. For smoke, these shapes can easily resemble its shape. However, for fire, it becomes more challenging since fire has very detailed and complex shapes. Therefore, it is necessary to do some post-processing on these heatmaps as detailed in the following Section.



(a) Original Image 1 **(b)** CAM heatmap 1 **(c)** Binary mask 1

(d) Original Image 2 **(e)** CAM heatmap 2 **(f)** Binary mask 2

(g) Original Image 3 **(h)** CAM heatmap 3 **(i)** Binary mask 3

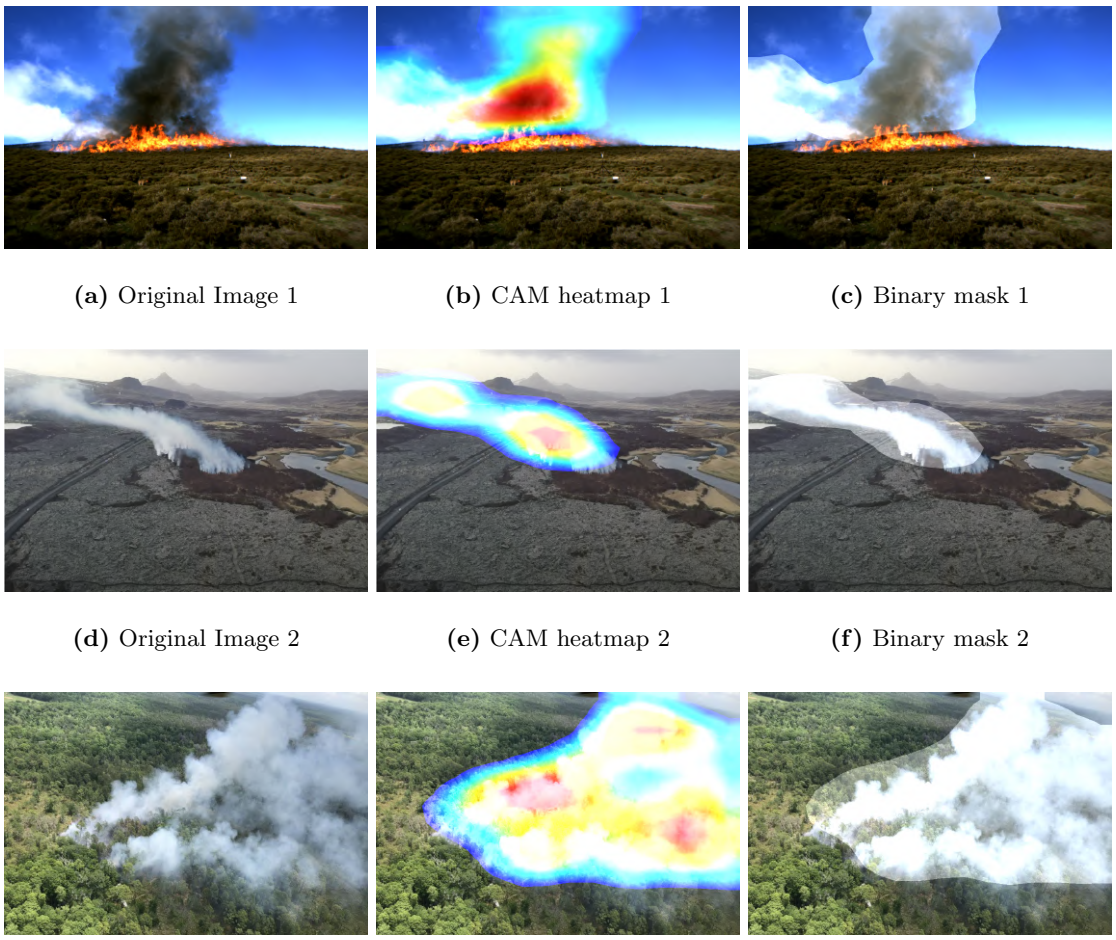
Figure 4.10: CAM output heatmaps fire examples

4.3.3 Post-processing Stage

In order to address the lack of detail in the masks produced by CAM, there was the need to do some post-processing. These masks hardly resemble the shape that fire and smoke take, especially fire, since it has a very detailed and irregular shape, which in turn is one of the reasons why it is so expensive to create pixel-level fire labels.

For these reasons, the CRF with fully connected nodes [41] was used to transform the binary masks created by CAM (with little detail), into masks that could actually resemble the shapes needed with well-defined boundaries and with much more detail. To achieve this, the CRF minimizes an energy function as in Equation (3.2). It is divided in two potentials, the unary described by Equation (3.3) and the pairwise described by Equation (3.4).

The unary potential, as in Equation (3.3), is responsible for assigning a cost to each pixel, according



(a) Original Image 1 **(b)** CAM heatmap 1 **(c)** Binary mask 1

(d) Original Image 2 **(e)** CAM heatmap 2 **(f)** Binary mask 2

(g) Original Image 3 **(h)** CAM heatmap 3 **(i)** Binary mask 3

Figure 4.11: CAM output heatmaps smoke examples

to its probability in the CAM mask. As the CRF takes information from all the pixels in the image, it is needed a unary potential for the foreground as well as the background and for that reason, every pixel in the CAM mask that is not considered to be positive for fire/smoke is considered to be background. So, the two possible states for each pixel are fire/smoke and background. The background mask $M_b(x, y)$ is the opposite of the foreground mask $M_f(x, y)$ as:

$$M_b(x, y) = 1 - M_f(x, y), \tag{4.9}$$

In the foreground mask, every positive pixel is set to 0.8 since CAM masks are not extremely precise.

The pairwise potential will then be responsible for analyzing each pixel and compare it with the neighbours in terms of nearness and similarity. It is a weighted combination of Gaussian kernels and it is divided into two kernels, as described in Equation (3.5), the appearance kernel and the smoothness

kernel. The $w^{(1)}$ and $w^{(2)}$ are set to assign an importance for each kernel.

The first kernel, the appearance kernel, uses the information of the pixel colour (RGB values) and the distance to their neighbours to assign the cost of the pixel. It is divided into two parts. One that is responsible to analyse the degree of nearness controlled by the spatial standard deviation θ_α and for both fire and smoke scenarios, since CAM binary masks cannot correctly delimit their boundaries, it should be set to a high value. Therefore, each pixel is compared with a wide range of pixels around it, allowing to create a segmentation of the whole area of the fire or smoke. The second part is responsible to analyse the degree of similarity controlled by the colour standard deviation θ_β. For the fire situation and taking into account that fire has a very specific and limited colour range, the θ_β should be set to a low value. Therefore, only pixels with very identical colour ranges are considered to be in the same class. Regarding smoke, the θ_β cannot be as low as fire since smoke colours can range a little more depending on the type of material burning, but it must be set low, too.

The second kernel, smoothness kernel, uses pixel proximity to remove small isolated regions and give the mask a much sharper boundary, controlled by the smoothness standard deviation θ_γ. For both situations, the value should be chosen to remove some miss-detected areas and give the fire and smoke the correct boundary limits. For both situations, a total of 5 inference steps are performed to get the final mask. Table 4.7 list the best parameters achieved to produce the final masks. To understand better the influence of each parameter in the CRF final output and the number of inference steps needed to create a good mask, different parameters were tested, as presented in the Experiment E in Section 5.5. Figure 4.12 and Figure 4.13 shows examples of the overall approach of the CRF using the best parameters for fire and smoke respectively .

Table 4.7: CRF best parameters

Parameter	**Fire**	**Smoke**
$w^{(1)}$	10	8
θ_α	250	100
θ_β	10	5
$w^{(2)}$	5	5
θ_γ	20	10
Iterations	5	5

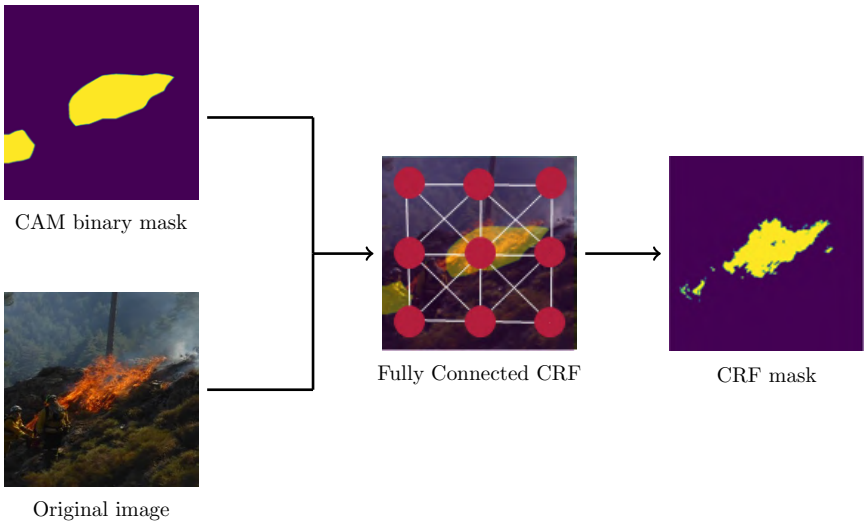


Figure 4.12: Fully connected CRF overall approach for fire

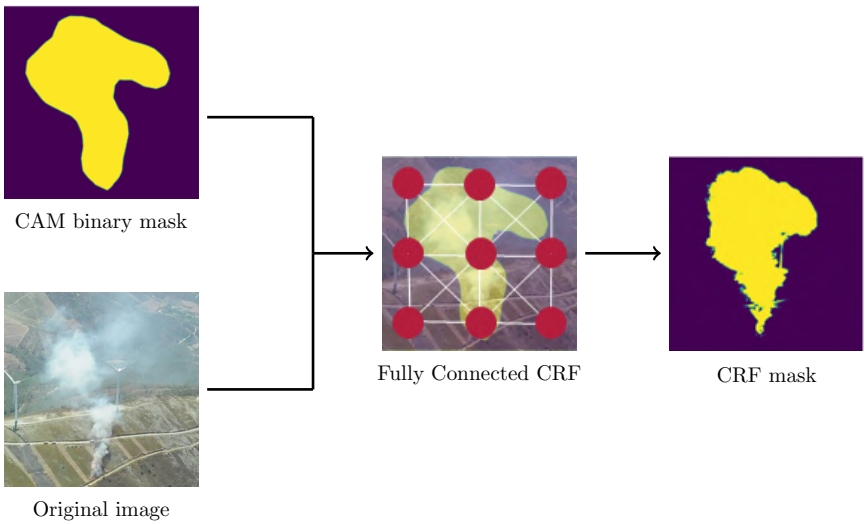


Figure 4.13: Fully connected CRF overall approach for smoke

4.4 Work Setup

To develop, validate and test the proposed approach two machines were used as described in Table 4.8. The frameworks used to develop the ANN were Keras and Tensorflow.

Table 4.8: System setup

Component	Machine 1	Machine 2
Processor	Intel i7-9700 3.00GHz	Intel i7-8700 @ 3.20GHz
GPU	Nvidia GeForce GTX 2080	Nvidia GeForce GTX 1070
GPU memory	12 Gb	8 Gb
RAM	64Gb	64Gb

4.5 Experiments list

The following list describes the experiments created to validate and test the proposal system and each of its' components individually.

- **Experiment A - Comparing the proposed approach with a fully-supervised one**

  The goal of this experiment is to state the pros and cons of using a weakly supervised approach versus a fully supervised. One wants to evaluate the gain or loss in performance with respect to the labelling effort.

- **Experiment B - Assesing the Classification Model**

  In this experiment, only the performance of the classification stage will be evaluated. One wants to test the overall robustness of the model with special attention to FP which can be very adverse in a forest fires situation.

- **Experiment C - Study the influence of negative feature maps on CAM**

  This experiment will evaluate if the weighting approach used in the CAM method is being used in favour of the segmentation performance. It will be studied, for both models, if the subtraction of negatively weighted feature maps is removing relevant information.

- **Experiment D - Search the best thresholding technique for CAM heatmaps**

  This experiment will analyze which is the technique to threshold the probability heatmaps from CAM, that lead to the top performance.

- **Experiment E - Search for CRF optimized parameters**

  In this experiment, the parameters of the CRF method will be varied to understand the influence of each one of them and what is the best combination that leads to the best optimization of the segmentation masks.

- **Experiment F - Evaluating the CRF influence**

  In this experiment, it will be evaluated how much the CRF improves the segmentation masks on the post-processing stage.

- **Experiment G - Comparing the Masks Areas**

  This experiment will serve to compare the masks in the different stages of the approach with the GT ones. Moreover, it will also be useful to understand the CRF reduction factor and how close the areas of the CRF masks get to the GT ones.

5

Experimental Results

Contents

This chapter details the experimental procedures to validate and test the proposed system. It is divided in each of the experiments introduced in the previous section. It is presented not only the results of the whole system but also the intermediate results on each stage: classification, weakly-supervised segmentation and post-processing. A comparison between the proposed method and two fully supervised ones evaluating the strengths and weaknesses of both approaches. It is also described some experiments done to optimize the different stages.

5.1 Experiment A - Comparing the proposed approach with a fully-supervised one

In this experiment, are compared two fully supervised segmentation methods (Method 1 and Method 2) and the proposed one. For Method 1 is done an extensive comparison on both fire and smoke segmentation while for Method 2 only the metrics for the fire case. Both models were developed in-house by members of the Firefront team using similar datasets.

- **Method 1** [49] - Originally the system was composed by three components. The first block analyzes the image and scales down to the size of the network input, for detecting fire/smoke using a classification model (SqueezeNet [50]). If the classification is positive the scaled image is fed to a segmentation network, otherwise the unscaled image divided in 4 patches and the same process is repeated for each patch. When an image/patch is classified as fire/smoke, it goes to the segmentation network (U-Net [51]) to create a segmentation mask with the regions of the image that contains fire/smoke. The obtained masks are then stitched in the right places to obtain the overall segmented image. The objective of Method 1 is to be able to detect fire/smoke in high resolution images even when the fire/smoke regions are just a few pixels. However, because the proposed method only uses images scaled to the network input size, the comparison will be done using a simpler version of the method without the recursive patch subdivision. This network was trained with a dataset fully annotated at the pixel level.

- **Method 2** - The second method consists in a Deeplab v3+ network [52] applied to the fire detection and was also trained with a dataset fully annotated at the pixel level. For this second method we only had access to the metrics values, we did not have access to the segmentation masks.

Both methods were tested using the Pixel-level dataset described in Section 4.1. In order to compare them, the average mIoU was computed.

For fire, the performance is shown in Table 5.1. It can be seen that both fully-supervised methods outperform the proposed one. However, it must be taken into account the effort that the authors had to put to create the strong supervision on their training examples versus the effort of creating weakly

supervised supervision. As concluded in Section 1.1 with the study done, creating pixel-level masks can take 50 times more time than a image-level mask. So, it is natural that the performance of fully-supervised methods is better than the weakly-supervised ones. Despite the difference in mIoU, the proposed method can also achieve considerably good results as it can be seen on Figure 5.1. The fully-supervised masks were obtained using Method 1 and the proposal masks are the output of the proposal method. In Figure 5.1 is noticeable that the proposed method achieves very good final segmentation mask and sometimes more identical to the GT masks than the ones provided by Method 1. The proposal masks can even better represent the small details in the fire shape when compared with the fully-supervised segmentation performed in the entire image. However, from the standard deviation results in Table 5.1 it is concluded that the proposed method has a higher oscillation in the results compared to Method 1. In Figure 5.3 there are some examples where this is illustrated. The proposed methods sometimes has some discrepancies resulting in a degradation of the segmentation mask, while Method 1 is more coherent. This is also reflected in the average mIoU values as these discrepancies in the proposed method results result in lower values of this metric. In summary, the proposed weakly-supervised method can highly compete with the fully-supervised ones with good fire segmentation results despite some small discrepancies.

Table 5.1: Performance of the tested fire models

Method	Approach	mIoU	Standard Dev.
Method 1	fully-supervised	0.856	0.073
Method 2	fully-supervised	0.902	-
Proposed	weakly-supervised	0.735	0.142

For smoke, the proposed approach will be only compared with Method 1 as Method 2 was not trained to detect smoke. The average mIoU results are shown in Table 5.2. The results show that the proposed method performs on par with the fully-supervised one, achieving a similar values of mIoU and standard deviation. As concluded before, the proposed method can easily resemble the round and soft margins of the smoke zones. Figure 5.2 represent some results where the proposed method outperforms the fully-supervised one. Not only the proposed mask can represent the outer shape of the smoke but can also outline objects that are inside the smoke area. The fully-supervised masks are more conservative without much detailed margins. However, as expected, there are also some not fully successful examples as illustrated in 5.4. These examples occur in images with areas with very similar colours to smoke, like clouds, or when it is not clear the separation with common co-occurring objects and zones. In summary, the proposed weakly-supervised method can perform almost as good as a fully-supervised segmentation method, with the advantage that no pixel-level masks are needed which in the case of smoke can be very ambiguous since smoke does not have sharp boundaries and can sometimes be very dim.

This experiment has shown that even by only using annotations at the image-level to train the proposed method, it is possible to compete with methods that uses annotations at the pixel level. The

Table 5.2: Performance of the tested smoke models

Method	**Approach**	**mIoU**	**Stantard Dev.**
Method 1	fully-supervised	0.771	0.157
Proposed	weakly-supervised	0.760	0.149

tedious and expensive process of creating pixel-level labels is not completely reflected in the segmentation results, especially for smoke, where the proposed method performs as well. When considering the trade-off between segmentation performance and the expensiveness of the creation of pixel-level labels the proposed smoke segmentation method is the clear winner, while the fire segmentation model also wins if the task at hand tolerates some distortions in the segmentation masks.

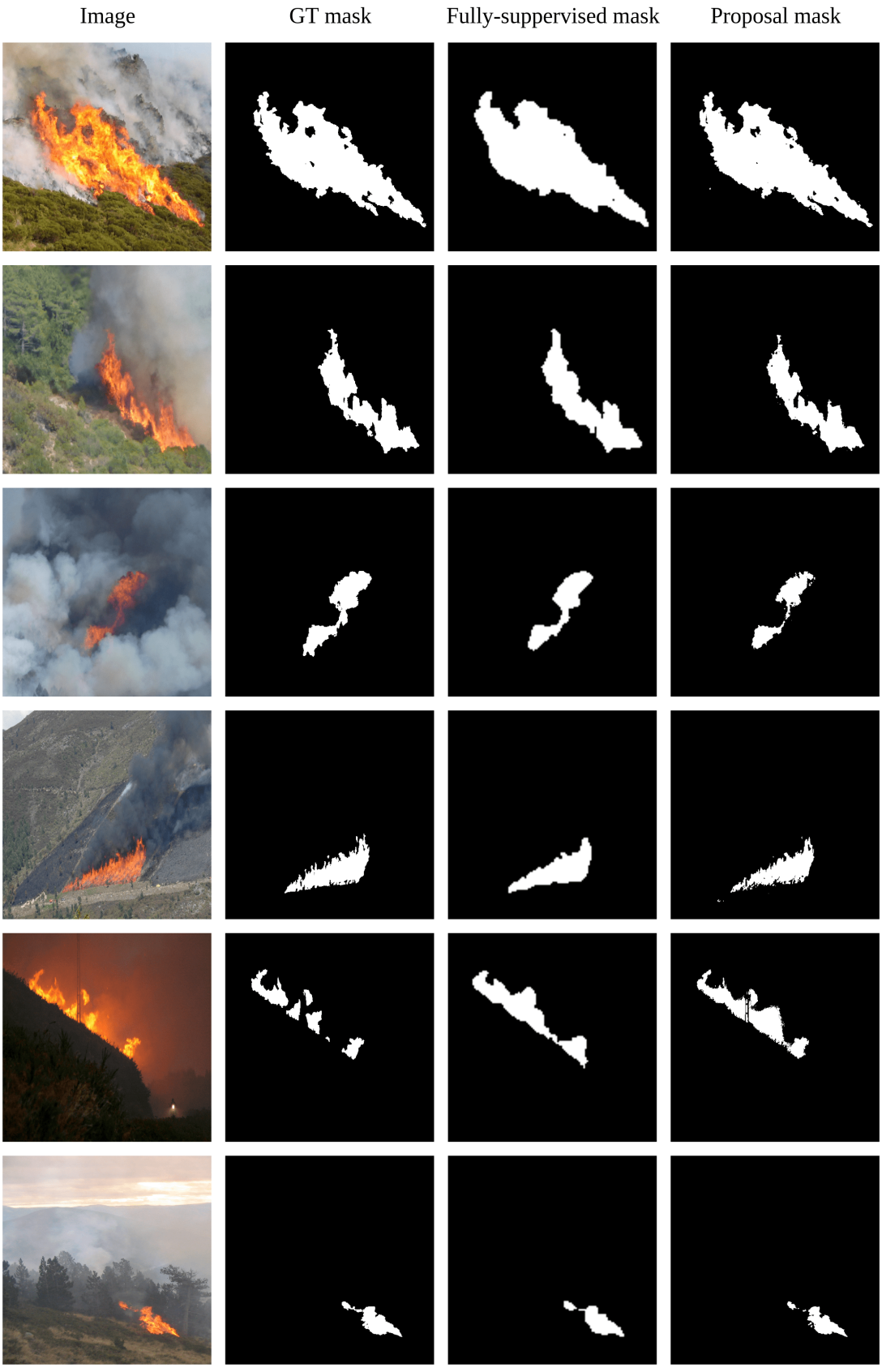


Figure 5.1: Fire masks comparison between a fully-supervised approach and the proposed approach.

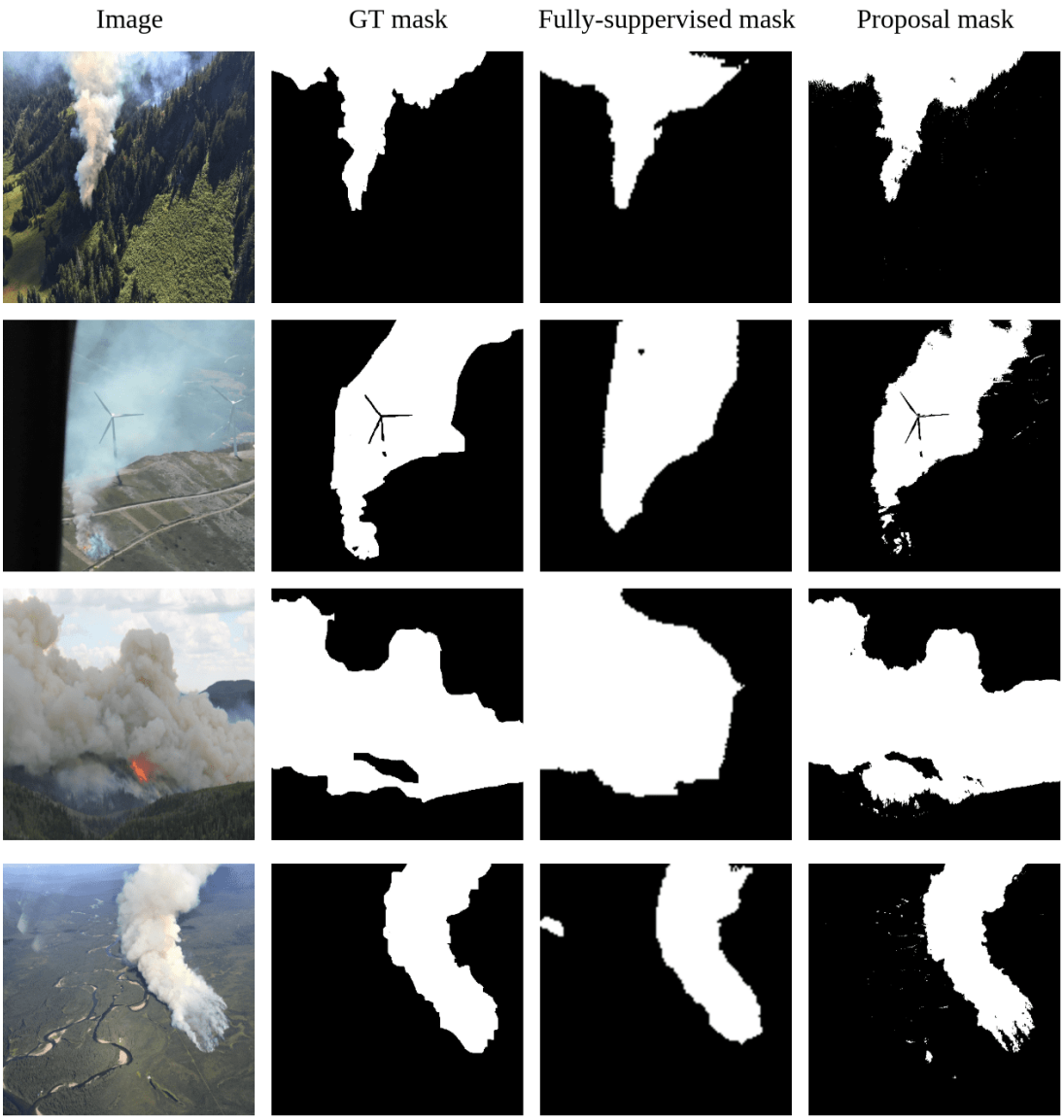


Figure 5.2: Smoke masks comparison between a fully-supervised approach and the proposed approach.

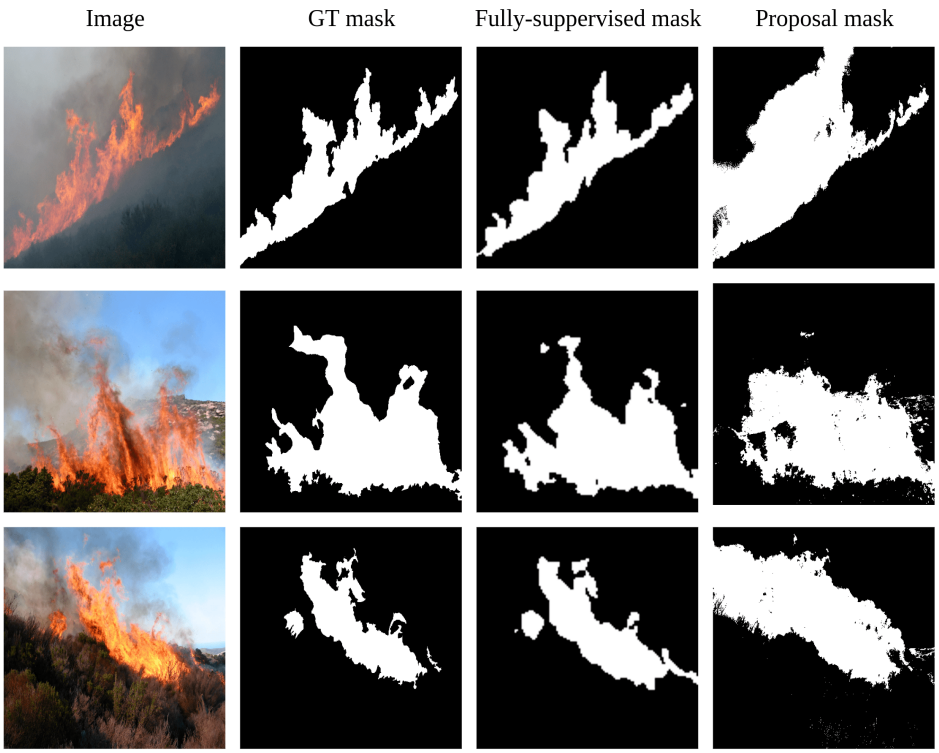


Figure 5.3: Fire masks comparison between a fully-supervised approach and the proposed approach with failure examples.

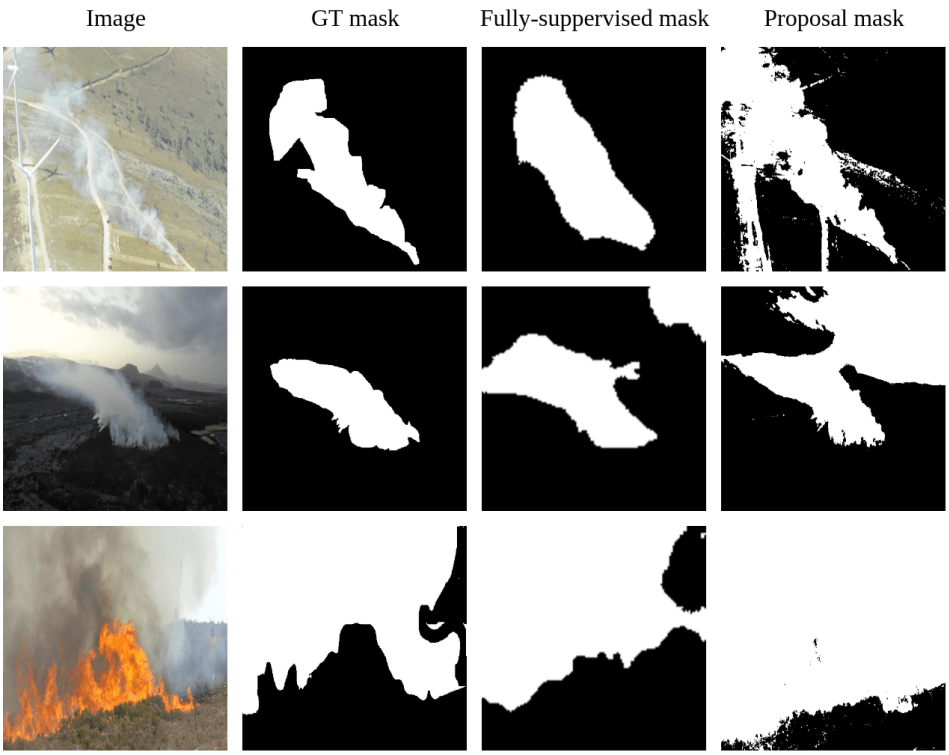


Figure 5.4: Smoke masks comparison between a fully-supervised approach and the proposed approach with failure examples.

5.2 Experiment B - Assessing the Classification Model

This experiment will evaluates the performance of both models regarding classification. For this stage it was used the dataset labelled at the image-level.

First, and as mentioned in Section 4.3.1 it is necessary to threshold the output probabilities of the network to classify a prediction as positive or negative. The thresholding is done as in Equation (4.5). To choose the one that produces the best result, the classification threshold was varied in the range $[0.4, 0.9]$ with steps of 0.1. The resulting accuracy in the validation set is plotted in Figure 5.5 for the different thresholds.

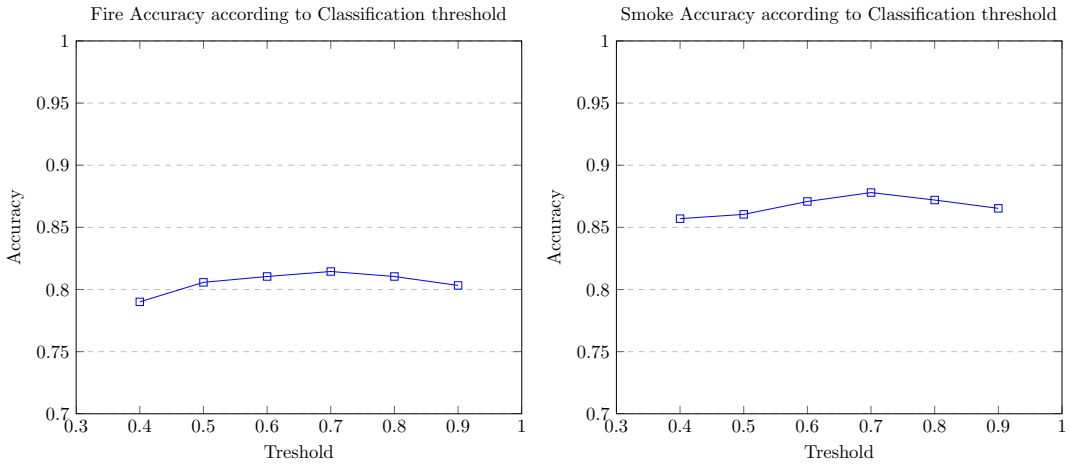


Figure 5.5: Fire and Smoke model tresholding and the corresponding Accuracy.

For both models the best threshold was 0.7 leading to an accuracy of 0.814 and 0.878 for the fire and smoke respectively. Table 5.3 shows the resulting accuracy, precision and recall in the test set with the best thresholds in the test set.

Table 5.3: Classification Performance

Metric	Fire Model	Smoke model
Accuracy	0.854	0.911
Precision	0.834	0.937
Recall	0.901	0.907

From table 5.3, it can be observed that the fire classifier does not have a very high accuracy nor precision value, but it presents a good recall value. This means that the number of FN is small, i.e., there are few predictions in which there was indeed fire but the model did not predict it. This is a good in a real situation when one does not want to neglect the presence of fire. On the other hand, the precision value is lower, which means that there was a considerable number of FP, i.e., there are images that effectively do not have fire but are being classified as fire. The vast majority of these images are smoke images without fire, which demonstrates once again the high correlation between fire and smoke and the challenge that is

to separate both by using labels at the image level. For the smoke situation, all the metrics are slightly higher. Nevertheless, it is possible to perceive the same correlation situation as in fire. In this case, the recall value is a little lower than the precision value since the number of FN is higher than the number of FP. In other words, there are more cases where there was no smoke but the model predicted that there was smoke than cases where there was smoke and the model predicted that there was not.

The models' overall classification performance is good even though the smoke model is slightly superior to the fire one. As expected, it is not easy for the network to correctly extract the class-specific features which make fire and smoke classification at the image-level quite challenging.

5.3 Experiment C - Study the influence of negative feature maps on CAM

In this experiment, it is studied how to optimize the identification of the regions of fire and smoke using the CAM algorithm. For both models, it will be analyzed to what extent the addition of negatively weighted feature maps affects the heatmaps.

According to Equation (3.1) the final CAM heatmaps is a weighted sum of the feature maps in the last convolutional layer. These weights range from -1 to 1, where a -1 indicates that the corresponding feature is not favourable for the output class classification and a 1 represents a favourable feature. This way, all the positive feature maps should be added and all the negative ones should be subtracted, resulting in a final heatmap highlighting the region associated with the predicted class. However, as mentioned before, the model is only trained to perform a classification at the image level, so the feature-maps might not always represent class-specific features. Ideally, the negative weights represent regions outside of the object like background. Nonetheless, a negative weight could also represent a feature map of a not so relevant part of the class and so, this way, this region would be removed from the final heatmap. Therefore, the interpretation of the positive and negative values on the weights could not be so straightforward as mentioned.

To understand if the standard CAM approach is missing information about regions relevant to the output class, a new heatmap is created where all the negative weights will be neglected [48]. This means that a negative feature-map will not take part in the sum as in Equation (4.6). So, there will be no feature map subtractions, only additions. To compare between using the negative weights or not, a set of 200 images from the Pixel-level dataset were used to compute the mIoU of the CAM raw output (without post-processing via CRF). The probabilistic heatmaps are converted into binary mask according to Equation (4.7) and Equation (4.8). For the fire model, the α is set to 0.5 and for the smoke, one is set to 0.2. The results are shown in Table 5.4.

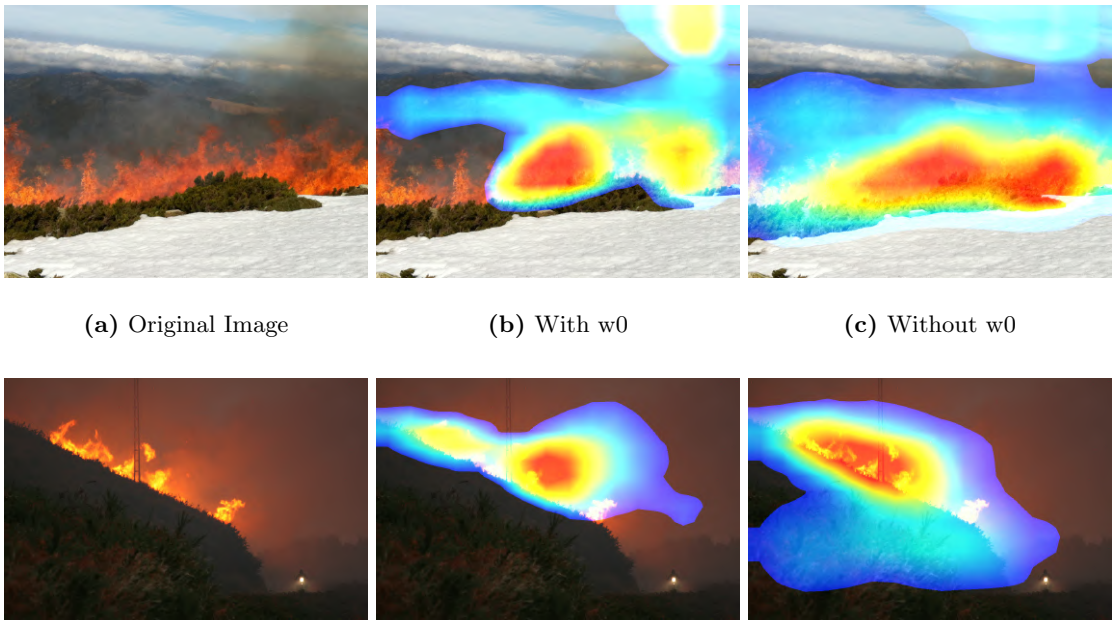
For the fire case, the improvement is quite noticeable when not using feature maps with negative weight

Table 5.4: mIoU values for both models with and without the use of negative weighted feature maps

Model	mIoU	
	w/ negative weigths	**w/o negative weigths**
Fire model	0.4367	0.6076
Smoke model	0.7036	0.5693

as illustrated in Figure 5.6. Ideally, negative weight feature maps should be enabled in background areas. However, this is not verified because one is working with image-level classification. This shows that these negative feature maps correspond to fire zones, especially to not-so-relevant areas. By neglecting the negative weights, it is ensured that the fire location is extended to almost the entire fire area, even in less significant areas.

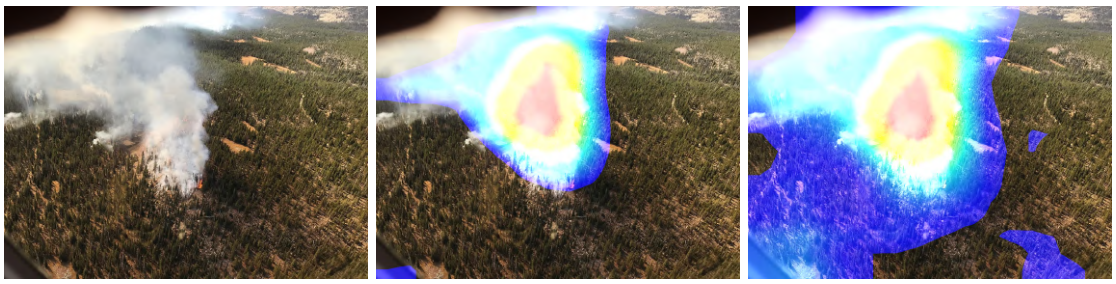
For the smoke case, there is no improvement in not using feature maps with negative weights as illustrated in Figure 5.7. This suggests that, unlike the fire model, feature maps with negative weights are being associated with background zones and not smoke zones. Thus, their subtraction is beneficial for the construction of the heatmap.



(a) Original Image **(b)** With w0 **(c)** Without w0

(d) Original Image **(e)** With w0 **(f)** Without w0

Figure 5.6: CAM output heatmaps fire examples



(a) Original Image **(b)** With w0 **(c)** Without w0

Figure 5.7: CAM output heatmaps smoke examples

5.4 Experiment D - Search the best thresholding technique for CAM heatmaps

In this experiment, two different techniques are compared to threshold the probabilistic heatmaps produced by the CAM algorithm. In the standard CAM approach, the segmentation masks are created by thresholding the heatmap according to its max value in Equation (4.7). However, [48] suggested a better method using probability distributions in the heatmap:

$$\tau = \alpha \; perc_i(H), \tag{5.1}$$

where α is still a real between 0 and 1, H is the probabilistic heatmap, and $perc_i$ is the $i-th$ percentile. So, the heatmap will be thresholded according to the $i-th$ percentile of the probabilistic distribution in the heatmap. The idea is to attenuate situations in which a high activation largely overlaps in a series of feature maps creating a probabilistic zone with high maximum value. In this situation the original thresholding method will create a binary mask with a small region of the object to detect, discarding the remaining extent of it.

Both thresholding approaches were tested in a set of 200 images of the Pixel-level dataset, using different values for α.

Starting with the max thresholding approach, the α was ranged, for the fire case, in $[0.3, 0.8]$ with steps of 0.1 and for the smoke in $[0.1, 0.6]$ with steps of 0.1. The mIoU and the corresponding standard deviation were computed and the results are shown in Table 5.5 for both models.

For fire, the α value that presented a better results was 0.6 with an mIoU of 0.6076 and a standard deviation of 0.1157. For smoke, the best α was 0.2 with an mIoU of 0.7036 and a standard deviation of 0.1214.

Then, for the percentile thresholding approach, the percentiles ranged in $[80, 95]$ with steps of 5 and the α ranged, for the fire case, in $[0.5, 0.9]$ and for the smoke in $[0.1, 0.5]$ both with steps of 0.1. Similar

Table 5.5: Max tresholding for fire and smoke

Fire			Smoke		
α	**mIoU**	**St. Dev.**	α	**mIoU**	**St. Dev.**
0.3	0.4233	0.1160	0.1	0.6920	0.1247
0.4	0.5146	0.1119	**0.2**	**0.7036**	**0.1214**
0.5	0.5869	0.1077	0.3	0.6581	0.1283
0.6	**0.6076**	**0.1157**	0.4	0.6043	0.1360
0.7	0.5540	0.1259	0.5	0.5457	0.1390
0.8	0.5080	0.1234	0.6	0.4896	0.1366

Table 5.6: Percentile tresholding for fire and smoke

Fire				Smoke			
α	**Perc.**	**mIoU**	**St. Dev.**	α	**Perc**	**mIoU**	**St. Dev.**
0.5	80	0.489	0.115	0.1	80	0.659	0.142
	85	0.519	0.115		85	0.665	0.139
	90	0.554	0.117		90	0.673	0.136
	95	0.578	0.120		95	0.6822	0.1317
0.6	80	0.492	0.115	0.2	80	0.673	0.136
	85	0.526	0.116		85	0.682	0.132
	90	0.560	0.117		90	0.689	0.128
	95	**0.590**	**0.122**		**95**	**0.696**	**0.123**
0.7	80	0.520	0.114	0.3	80	0.680	0.132
	85	0.550	0.115		85	0.686	0.128
	90	0.574	0.116		90	0.690	0.124
	95	0.588	0.121		95	0.6895	0.122
0.8	80	0.539	0.113	0.4	80	0.678	0.129
	85	0.562	0.114		85	0.681	0.126
	90	0.577	0.117		90	0.679	0.124
	95	0.576	0.121		95	0.6671	0.125
0.9	80	0.552	0.112	0.5	80	0.671	0.127
	85	0.567	0.114		85	0.669	0.125
	90	0.573	0.120		90	0.659	0.127
	95	0.557	0.131		95	0.633	0.128

metrics to the max thresholding approach were computed and the results are shown in Table 5.6 for both models.

For fire, the parameters that lead to better results were an α of 0.6 and the 95^{th} percentile with an mIoU of 0.590 and a standard deviation of 0.122. For smoke, the best results were an α of 0.2 and the 95^{th} percentile resulting in an mIoU of 0.696 and a standard deviation of 0.123.

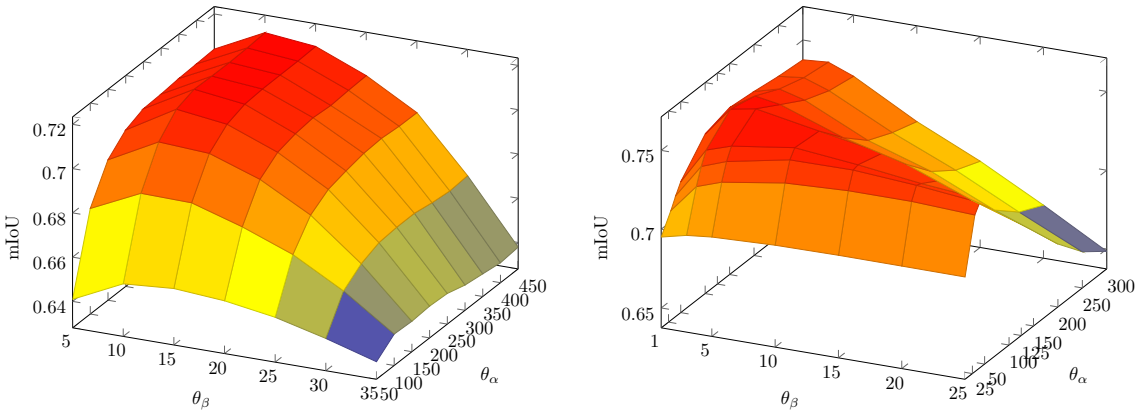
It is observed that the results for the percentile approach converge to the higher percentile value, being that a 100^{th} percentile would correspond to the same as using the max approach. Additionally, for both approaches the best alpha value is equal. For this α value and comparing the two approaches, it turns out that the max approach produces a higher mIoU value with a lower standard deviation for both models. In conclusion, it is not advantageous to use the thresholding method using the percentiles

of the probability distributions in the CAM heatmap.

5.5 Experiment E - Search for CRF optimized parameters

In this experiment, is one an exhaustive search on all the parameters that compose the CRF algorithm to optimize the CAM binary masks and consequently get a better final segmentation mask.

The post-processing phase consists of using the CRF algorithm with fully-connected nodes. As defined in Equation (3.4), this method depends on some parameters. The search was performed by maximizing the mIoU of the method in a set of 200 images of the Pixel-level dataset. We started by varying the various parameters with a rough discretization to understand the orders of magnitude and range of use. The number of inference steps started on 15, so that the evolution per step could be analyzed for a significant number of steps. Next, the weights of each kernel of Equation (3.5), defined by w^1 and w^2, were set. The first kernel incorporates a term that is colour dependent and he second kernel will be responsible for giving the detail and sharpness. For the fire case, we assign colour a bigger importance than sharpness, thus, w^1 was set to 10 and w^2 to 5. For the smoke case, the first kernel is not so important since smoke is not so easily characterized by its colour while the second kernel will play an important role in defining the edges of the smoke. Therefore, w^1 is set to 8 and w^2 to 5. Finally, an optimum value for θ_y was set by the same logic previously explained concerning the second kernel. That is, a higher θ_y is required for the fire case than for the smoke case. So, θ_y was set to 20 for fire and 10 for smoke. Finally, an exhaustive search for the parameters of the appearance kernel, θ_α and θ_β, was performed. For fire, θ_α was ranged in $[50, 450]$ with steps of 50 and θ_β was ranged in $[5, 35]$ with steps of 5 as illustrated in Figure 5.8(a). For smoke, θ_α was ranged in $[25, 300]$ with steps of 25 and θ_β was ranged in $[1, 25]$ with steps of 5 as illustrated in Figure 5.8(b).



(a) Fire **(b)** Smoke

Figure 5.8: θ_α and θ_β variation for fire and smoke.

Regarding the fire case, it is clearly evident the high dependence on the colour component controlled by the θ_β. Small variations in this parameter are directly reflected in the mIoU values, reaching an optimum value of 10. Keeping this parameter fixed and varying the θ_α there is an increase up to about 250, after witch it stagnates. The higher this parameter, the higher the standard deviation of the results. Therefore, the optimal parameters will be θ_α equal to 250 and θ_β equal to 10. Figure 5.9 also concludes this regarding θ_α. A value lower than 10 produces a mask containing small areas that do not correspond to fire. Above 10, the algorithm gets too selective and actually removes fire zones.

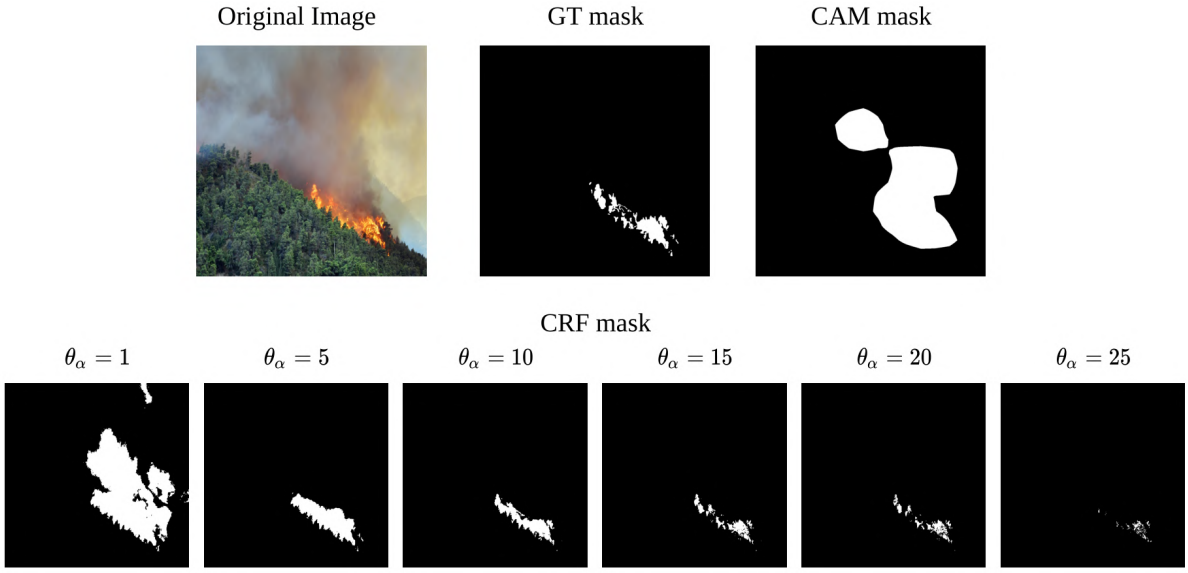


Figure 5.9: CRF mask evolution according to θ_α for a fire example

For the smoke case, the situation is identical but the parameters have swapped. Now the results are more sensitive to θ_α and less sensitive to θ_β. That makes sense since smoke does not have such a characteristic colour. The best parameters are θ_α equal to 100 and θ_β equal to 5. Figure 5.10 also concludes this regarding θ_β. A value lower than 100 produces a masks containing some unwanted areas and above it, it removes some smoke areas.

Finally, with the remaining parameters already configured, it was necessary to analyse the number of inference steps that the algorithm would need to converge to a good value of the energy equation (3.2) and consecutively a good segmentation mask. The higher the number of iterations the higher is the computational time. It was verified that, with this set of parameters and in most of the cases, the algorithm converges until the 5^{th} iteration, as it can be seen on Figure 5.11. Beyond that, the energy curve stays flatten and the changes were not visible. Figures 5.12 and 5.13 illustrate the algorithm iterations till the 5^{th} one.

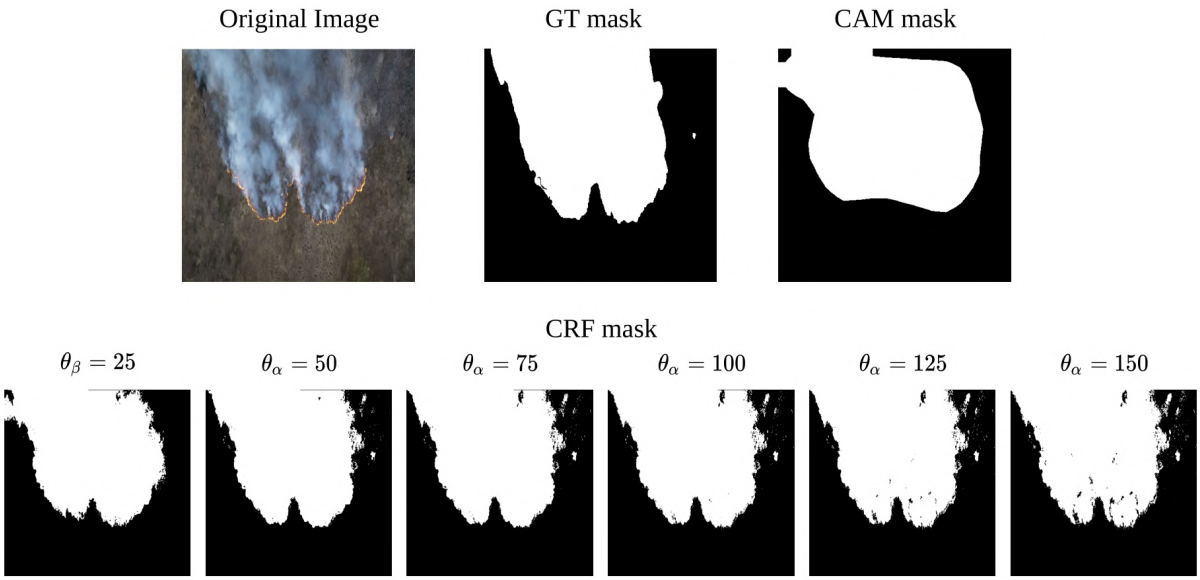
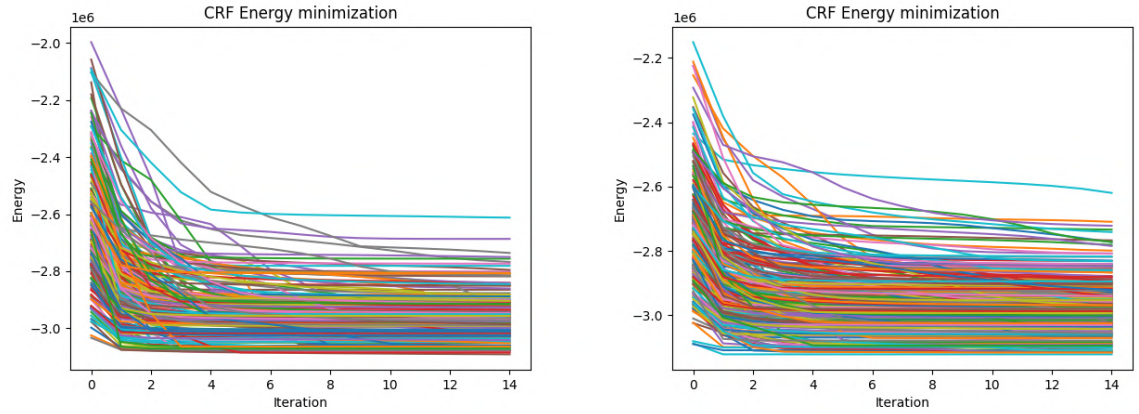


Figure 5.10: CRF mask evolution according to θ_β for a smoke example



(a) Fire

(b) Smoke

Figure 5.11: CRF energy convergence iteration by iteration.

5.6 Experiment F - Evaluating the CRF influence

This experiment evaluates how the CRF affects the system performance in the two phases of the pipeline: weakly supervised segmentation (WSSegm.) and post-processing (Post.Process.). The first stage is evaluated through the masks obtained by CAM while the second stage is evaluated through the masks obtained after the application of the CRF. It will also be evaluated the trade-off between the use of the post-processing stage and processing time. The results that follow were obtained using the Pixel-level dataset.

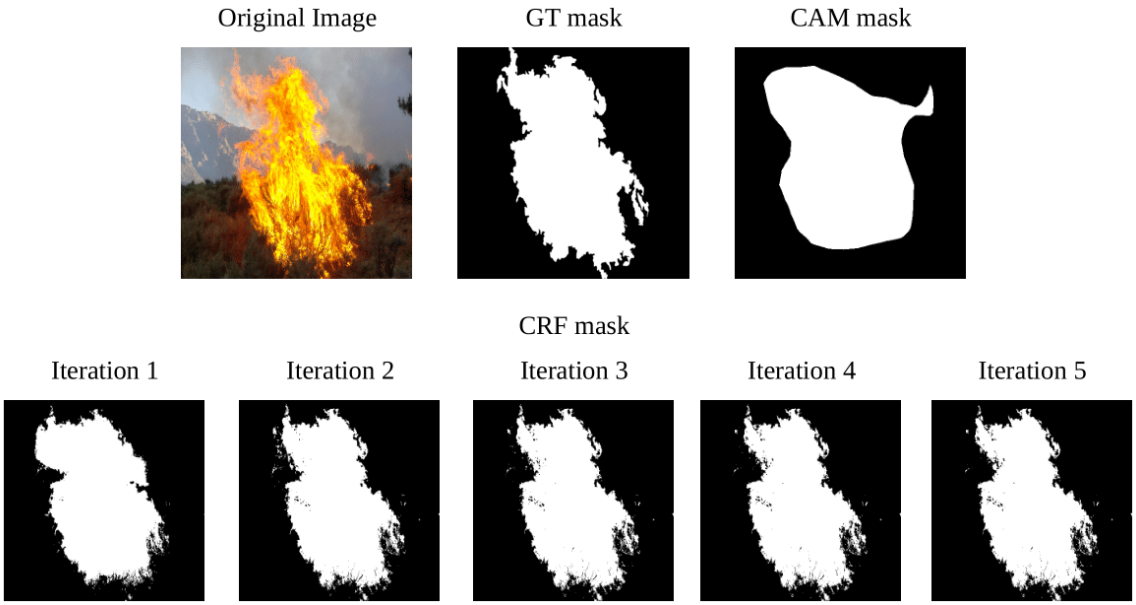


Figure 5.12: CRF mask iterations for a fire example

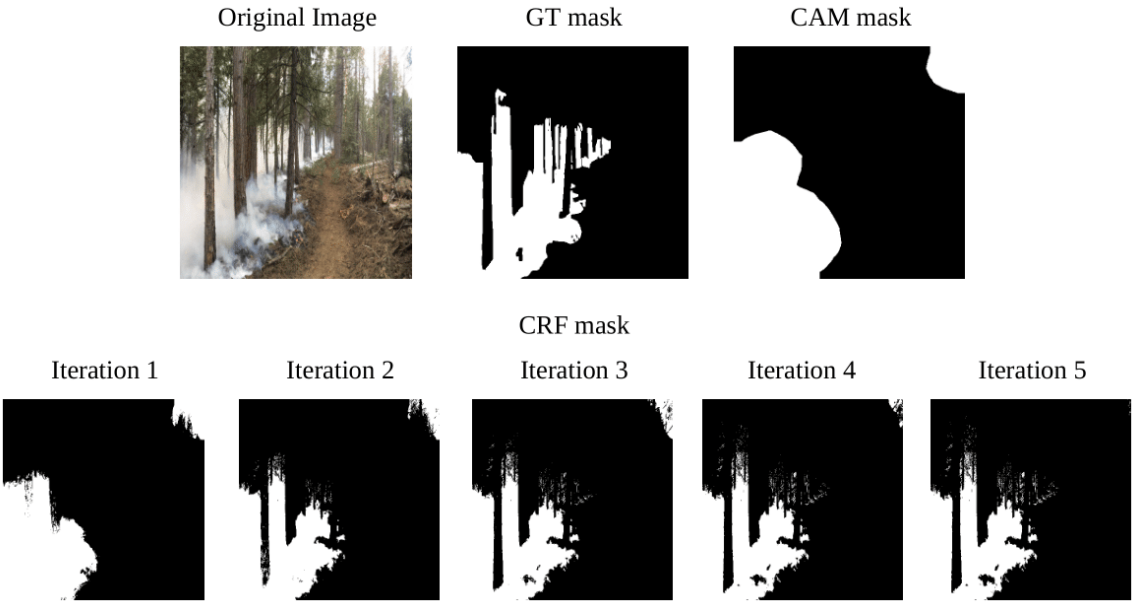


Figure 5.13: CRF mask iteration for a smoke example

Table 5.7 illustrates the performance of the two stages for both models in terms of the mIoU, the corresponding standard deviation and the processing time.

Firstly, one must highlight the good performance of weakly supervised segmentation (WSSegm) taking into account that the model was only trained for image classification at the image-level. It can be observed that the mIoU values for the smoke case are higher than the fire values. This can be explained by the fact that while the fire shape can be quite detailed, smoke usually has a non-detailed shape, sometimes

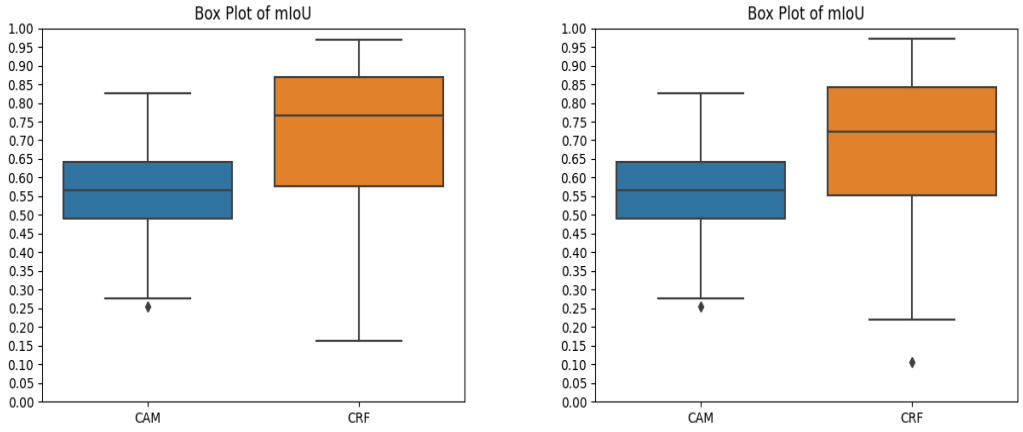
Table 5.7: Segmentation Performance in both stages

Model	Stage	mIoU	St.Dev.	Proc.Time (s)
Fire	WSSegm.	0.607	0.115	0.068
	Post.Process.	**0.735**	0.142	0.228
Smoke	WSSegm.	0.703	0.121	0.059
	Post.Process.	**0.760**	0.149	0.229

resembling blobs. The processing time in this phase is considerably low since CAM works with small mapping resolutions.

Secondly, after the application of the CRF, a great improvement in the mIoU is noticed. This improvement is much more significant for the fire case since it is necessary to add all the detail and sharpness of the fire shape. For fire, the improvement is about 20% while for smoke it is almost 10%. There is a slight increase in standard deviations but it is not comparable to the improvements in mIoU. Regarding the processing time, applying the 5 iterations of the CRF to each image took an average computation time of $230ms$ for both cases.

Furthermore, Figure 5.14 illustrates the distribution of mIoU in both stages. The improvement in the mIoU for both cases is quite remarkable, with the median and the IQR greatly improving. However, the increase in the standard deviation is noticeable, not only for the increase in the IQR size but also for the increase in the whiskers spread.



(a) Fire **(b)** Smoke

Figure 5.14: Box plot of the mIoU results

Figure 5.15 illustrates, for fire, some examples of the resulting masks in both stages of the approach as well as the GT mask. At first, one can conclude that CAM can correctly predict the location of fire, although the CAM masks are quite coarse, in the form of blobs. Thus, when comparing the CAM masks

with the GT masks, the mIoU difference is mostly due to lack of detail, rather than poor location. Then, after applying the CRF, the improvements in terms of detail and sharpness are highly notorious. The CRF can transform a coarse and blob-like mask roughly indicating the location of fire, into a mask very similar to the GT. The CRF takes great advantage of the fact that the fire has a very representative and limited colour space. The resulting masks are sometimes even more detailed than masks created on GT.

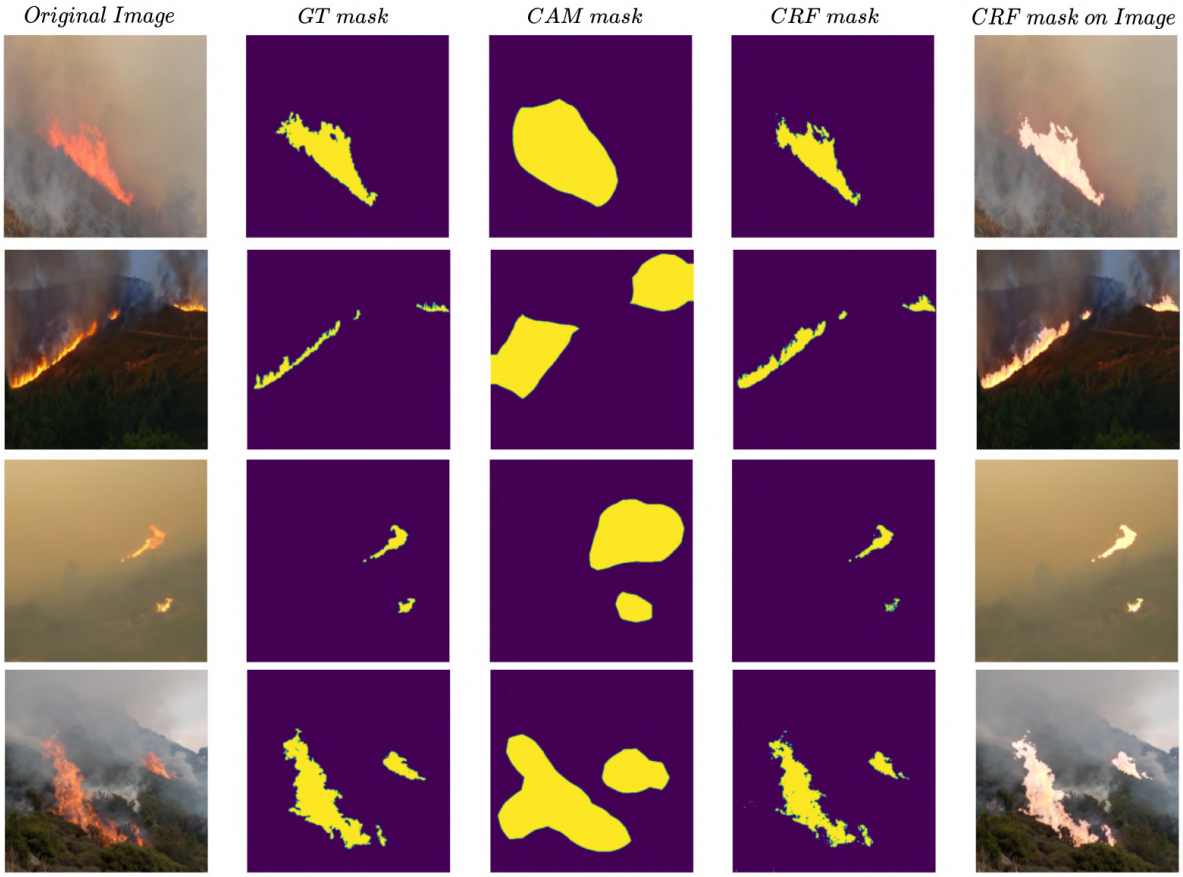


Figure 5.15: Fire masks in the different stages

Figure 5.16 illustrates, for the smoke case, some examples of the resulting masks in both stages of the approach as well as the GT masks. Considering that smoke does not have shape as precise as fire, it is easier to get a better segmentation mask using only CAM. This is also reflected in the mIoU values in Table 5.7 using only CAM, which are already considerably good. However, it is still necessary to use CRF to correctly delineate the smoke edges. This includes removing areas where CAM masks give overlay between smoke and other non-smoke areas, for example, fire. After applying the CRF it turns out that the masks are much more detailed and much more accurate. Similarly to what happened with the fire, the masks after the application of the CRF can sometimes be more detailed than the ones created by hand.

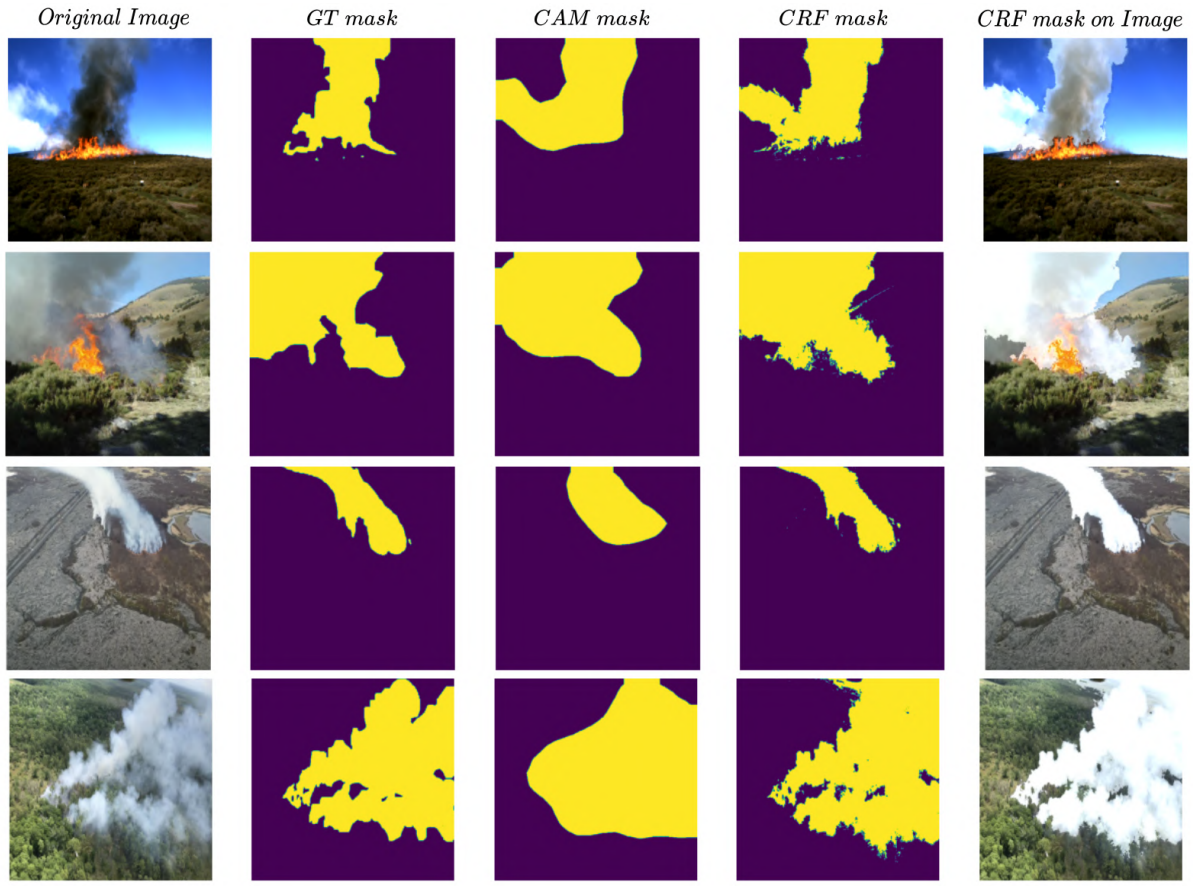


Figure 5.16: Smoke masks along the stages

Despite all the aforementioned advantages, the CRF is still totally dependent on the input CAM mask. So, when this mask gives an unreasonable segmentation, it can sometimes happen that the CRF converges to non-fire/non-smoke zones.

5.7 Experiment G - Comparing the Masks Areas

In this experiment, is made a detailed analysis of the size of the masks obtained in the two phases of the proposal n comparison with the GT using the Pixel-level dataset. In particular, we compare the areas of the three masks: CAM mask, CRF mask and GT mask.

For the comparison a series of ratios were created:

- **GT/Image** - GT mask area divided the number of pixels in the image.

$$GT/Image = \frac{\#NonZeroPixels(GT\,mask)}{\#Pixels(Image)} \tag{5.2}$$

- **CAM/Image** - CAM mask area divided by the number of pixels in the image.

$$CAM/Image = \frac{\#NonZeroPixels(CAM\,mask)}{\#Pixels(Image)} \tag{5.3}$$

- **CRF/Image** - CRF mask area divided by the number of pixels in the image.

$$CRF/Image = \frac{\#NonZeroPixels(CRF\,mask)}{\#Pixels(Image)} \tag{5.4}$$

- **CAM/GT** - CAM mask area divided by the GT mask area.

$$CAM/GT = \frac{\#NonZeroPixels(CAM\,mask)}{\#Pixels(GT\,mask)} \tag{5.5}$$

- **CRF/GT** - CRF mask area divided by the GT mask area.

$$CRF/GT = \frac{\#NonZeroPixels(CRF\,mask)}{\#Pixels(GT\,mask)} \tag{5.6}$$

- **CRF/CAM** - CRF mask area divided by the CAM mask area.

$$CRF/CAM = \frac{\#NonZeroPixels(CRF\,mask)}{\#Pixels(CAM\,mask)} \tag{5.7}$$

Using the first three ratios (GT/Image, CAM/Image and CRF/Image) the objective is to compare the percentages of fire/smoke occupation in the image. In Table 5.8 there are listed the average values for the three ratios for both fire and smoke, and on Figure 5.17 there are plotted the box plot of the ratios' distributions.

Starting with the fire situation in Figure 5.17(a), it can be seen that the dataset used has a somewhat reduced distribution of fire occupation in the images, with the vast majority of values being below 30%. However, there are still some outliers with high levels of occupancy. The occupancies in the CAM masks are considerably higher, the vast majority of them being above the median GT values. This is explained, once again, by the blob-like shapes covering the fire regions that CAM presents. Regarding the occupation in the CRF masks, it can be observed that it presents a distribution of values very close to the GT values with quite similar medians. Nevertheless, the presence of some outliers should be noted. These outliers are because these masks depend not only on the success of the CAM but also on the success of the CRF algorithm.

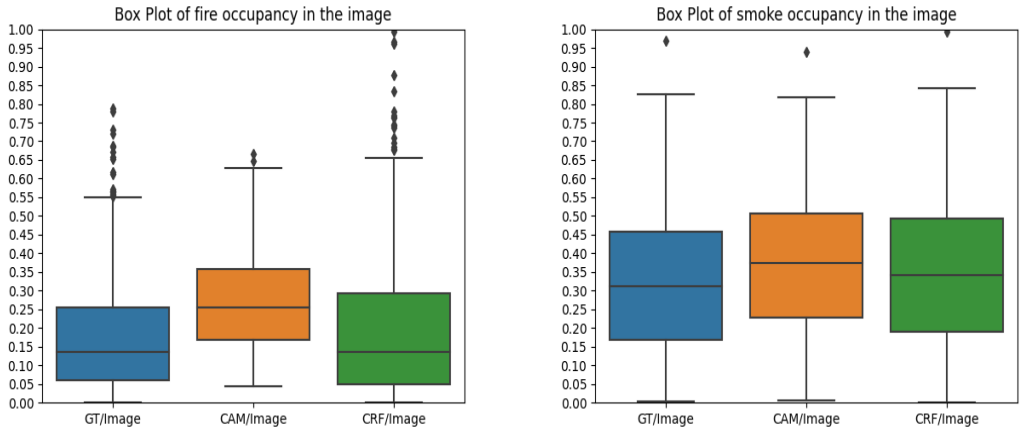
For the smoke situation in Figure 5.17(b), it can be seen that the images in the dataset have a rather dispersed occupancy distribution, with the majority of the images being around 35%. Similarly to what happened in the fire situation, it is observed that the CAM masks show a higher occupancy than the

GT values for the same reasons, but in this case, the discrepancy is not so pronounced. As in the fire situation, with the CRF masks, the smoke occupation distribution in the image becomes quite similar to the GT values. It can also be seen that the reduction of occupancy of the CAM masks is not so large.

On average, one can conclude that the fire/smoke occupancy in the CAM masks is considerably higher relative to GT values while for CRF masks the occupancy becomes quite similar, showing the great importance of the CRF post-processing step in the performance of the full pipeline.

Table 5.8: Ratio with the percentages of occupancy for fire and smoke

Model	Ratio	Average
Fire	GT/Image	0.179
	CAM/Image	0.269
	CRF/Image	0.193
Smoke	GT/Image	0.322
	CAM/Image	0.373
	CRF/Image	0.347

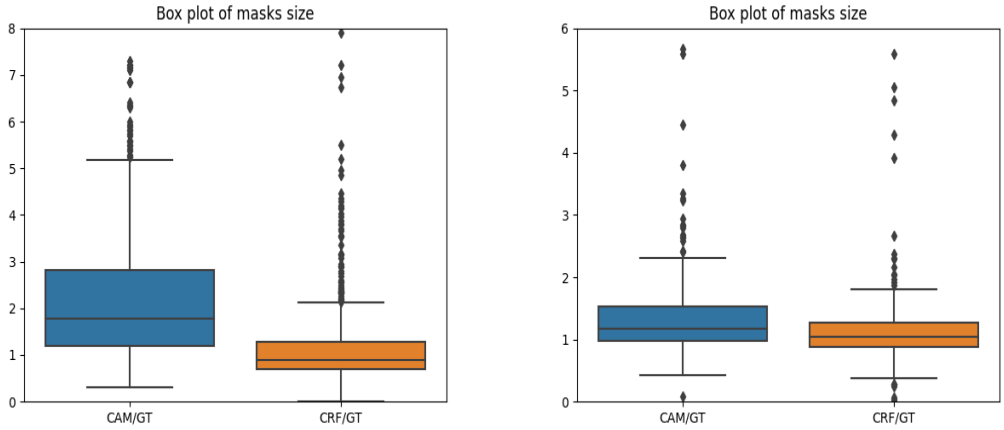


(a) Fire **(b)** Smoke

Figure 5.17: Box plot of the mIoU for the fire test set

To compare the masks directly with the GT ones, one can use the ratios CAM/GT and CRF/GT shown in Figure 5.18. The average values, in this case, are not fully representative of the distribution due to considerable number of outliers. These ratios reinforce what was previously reported. It is again highlighted the fact that the CAM masks are considerably bigger than those of GT. This is exacerbated in the fire situation where most of the masks are about two times bigger. As expected, the CRF to GT ratio is very close to 1 for both fire and smoke. It also demonstrates the great corrective factor of the CRF algorithm.

Finally, to quantify the CRF algorithm's reduction factor on the CAM masks, the CRF/CAM ratio



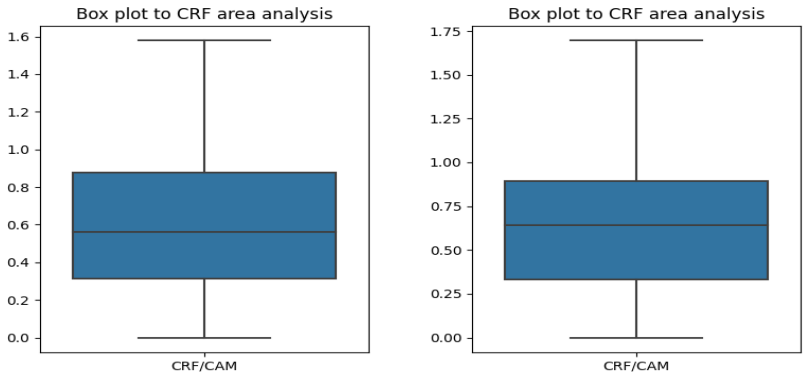
(a) Fire **(b)** Smoke

Figure 5.18: Fire and smoke occupancy in the image

is used. It is noticeable, both by the average values of Table 5.9 and the distributions in Figure 5.19, that the reduction in the size of the masks is quite significant. In the case of fire, this reduction is more pronounced, being that CRF masks have on average a size that is 60% of the size of the CAM masks. For smoke, this value is close to 84% since the CAM masks can almost define the shape of smoke.

Table 5.9: Ratio for comparison of CRF and CAM

Model	**Ratio**	**Average**
Fire	CRF/CAM	0.6016
Smoke	CRF/CAM	0.8463



(a) Fire **(b)** Smoke

Figure 5.19: Box plot of the mIoU for the fire test set

6

Conclusions and Future Work

Reaching the end of this thesis, it is time to draw some conclusions on the study and work developed, and state some proposals as future work to prevent failure cases and improve the system.

The lack of data in this field, namely the scarce amount of freely available datasets with pixel-level annotations together with the expensiveness and subjectivity of manually creating the annotations, has led to the creation of a system that relies only on weakly-supervised methods. The creation of labels at the image-level for these methods can be almost effortless and makes the process of gathering new images almost effortless.

The process of classifying and segmenting fire and smoke zones is already a great challenge using fully-supervised methods based on deep learning since the shape that fire and smoke zones take can be very irregular and sometimes very dim. When using weakly-supervised this challenge is even greater because it is then up to the network to understand which are the class-specific features of fire and smoke in the entire image. Thus, it was necessary to create a good and complex dataset with several different examples. It was important to have several examples of fire and smoke individually in order to not correlate both of them. Also, it was important to have various examples of negative images where none of are present, to distinguish common co-occurring zones, for example, vegetation.

The proposed method makes use of the powerful capabilities of CNNs on image classification and their ability to model patterns by finding representative class features. In addition, the method uses the base of classic methods for fire/smoke detection which is the very characteristic colour pattern.

As a result, the system developed has shown to be able to detect and segment fire and smoke zones in an accurate and precise way using only these type of methods. In particular, using the CAM method, it was proven that it is possible to train a classification model only with image-level labels and by extracting the features that the model uses for the classification prediction, one can construct a slightly rough heatmap highlighting the fire/smoke zones. Subsequently, by using an energy minimization algorithm, the CRF, it was possible to transform the rough heatmaps previously obtained into a considerably accurate segmentation mask of fire/smoke.

Even so, some limitations were noted using these methods. First, it was noted that as the model does classification in the whole image and the input image size must be small for computational reasons, images with very small zones of fire/smoke could not be detected. As a suggestion, the use of methods with a sliding window could be beneficial. Second, the several parameters in the post-processing stage are static and were tuned in a more generalized way resulting in some undesired situations. In future works, the tuning process may be done using a learning process, similar to the classification model, resulting in dynamic parameters that can adapt to each image. Third, the system presents some small oscillations in the performance results. In future studies, we suggest the use of semi-supervised methods where it can be combined both fully and weakly supervised methods. This way one could use the few datasets available annotated at the pixel level with the ease of gathering images to annotate at the image-level.

By combining both approaches it could be possible to develop a more robust and very accurate method for the fire and smoke segmentation.

The overall results show that when taking into account the heavy needs of a fully-supervised method, the proposed weakly-supervised system can strongly compete with them in terms of segmentation performance. For smoke, the proposed methods even achieve identical performance.

In conclusion, the thesis objectives can be considered fulfilled as the proposed system has proven to be able to accurately generate segmentation masks to detect fire and smoke at the pixel-level using only weakly-supervised methods at the image-level

A paper [53] from the initial stages of the work presented here was accepted and presented at the RECPAD 2020[1] conference.

With this work, we hope to represent a great contribution to the Firefront project and help it to support the brave firefight teams. We also hope that it will serve as motivation for future works in this area since the problem of wildfires is still a real situation and represents a catastrophe that seriously affects human beings and our planet.

[1] https://recpad2020.uevora.pt/

Bibliography

[1] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 2921–2929, 2016.

[2] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[3] N. C. Institute and F. (ICNF), "Incêndios rurais e área ardida – continente," Aug 2020. [Online]. Available: https://www.pordata.pt/Portugal/Inc%c3%aandios+rurais+e+%c3%a1rea+ardida+%e2%80%93+Continente-1192-9576

[4] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[5] S. Sharma, S. Sharma, and A. Athaiya, "Activation Functions in Neural Networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 04, no. 12, pp. 310–316, 2020.

[6] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016.

[7] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," *Neural Computation*, vol. 29, pp. 2352–2449, 2017.

[8] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization." *2017 IEEE international conference on computer vision (ICCV)*, pp. 3544–3553, 2017.

[9] C. Redondo-cabrera, M. Baptista-r, and J. L.-S. Roberto, "Learning to Exploit the Prior Network Knowledge for Weakly-Supervised Semantic Segmentation," *CoRR*, vol. abs/1804.04882, pp. 1–13, 2018.

[10] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. Huang, "Adversarial Complementary Learning for Weakly Supervised Object Localization," *CoRR*, vol. abs/1804.06962, pp. 1325–1334, 2018.

[11] Y.-t. Chang, Q. Wang, W.-C. Hung, R. Piramuthu, Y.-H. Tsai, and M.-H. Yang, "Mixup-CAM: Weakly-supervised Semantic: Segmentation via Uncertainty Regularization," *British Machine Vision Virtual Conference 2020*, pp. 1–13, 2020.

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, vol. 128, no. 2, pp. 618–626, 2017.

[13] P. Barmpoutis, P. Papaioannou, K. Dimitropoulos, and N. Grammalidis, "A review on early forest fire detection systems using optical remote sensing," *Sensors (Switzerland)*, vol. 20, no. 22, pp. 1–26, 2020.

[14] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho, and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," *IEEE Access*, vol. 6, no. c, pp. 18 174–18 183, 2018.

[15] C. Yuan, Y. Zhang, and Z. Liu, "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques," *Canadian Journal of Forest Research*, vol. 45, no. 7, pp. 783–792, 2015.

[16] G. Hristov, J. Raychev, D. Kinaneva, and P. Zahariev, "Emerging Methods for Early Detection of Forest Fires Using Unmanned Aerial Vehicles and Lorawan Sensor Networks," *2018 28th EAEEIE Annual Conference, EAEEIE 2018*, pp. 1–9, 2018.

[17] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, "Airborne optical and thermal remote sensing for wildfire detection and monitoring," *Sensors (Switzerland)*, vol. 16, no. 8, 2016.

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

[19] Y. Zhao, J. Ma, X. Li, and J. Zhang, "Saliency detection and deep learning-based wildfire identification in uav imagery," *Sensors (Switzerland)*, vol. 18, no. 3, 2018.

[20] T. Celik, "Fast and efficient method for fire detection using image processing," *ETRI Journal*, vol. 32, no. 6, pp. 881–890, 2010.

[21] H. Demirel and T. C-elik, "Fire detection in video sequences using a generic color model," *Fire safety journal*, vol. 44, pp. 147–158, 2009.

[22] D.-c. Wang, X. Cui, E. Park, C. Jin, and H. Kim, "Adaptive flame detection using randomness testing and robust features," *Fire Safety Journal*, vol. 55, pp. 116–125, 2013.

[23] D. Y. T. Chino, L. P. S. Avalhais, J. F. R. Jr, A. J. M. Traina, and S. Carlos, "BoWFire : Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis," *2015 28th SIBGRAPI conference on graphics, patterns and images*, 2015.

[24] A. E. Çetin, K. Dimitropoulos, B. Gouverneur, N. Grammalidis, O. Günay, Y. H. Habiboˇ, B. Uˇ, and S. Verstockt, "Video fire detection – Review," *Digital Signal Processing*, vol. 23, pp. 1827–1843, 2013.

[25] B. U. To¨reyin, D. Yigˇithan, G. Ugˇur, and A. E. C. Etin, "Computer vision based method for real-time fire and flame detection," *Pattern recognition letters*, vol. 27, pp. 49–58, 2006.

[26] M. Batista, B. Oliveira, P. Chaves, J. C. Ferreira, and T. Brandão, "Improved Real-time Wildfire Detection using a Surveillance System," *Proceedings of the World Congress on Engineering 2019 WCE 2019, Lecture Notes in Engineering and Computer Science*, vol. 0958, no. July, 2019.

[27] T. Toulouse, L. Rossi, M. Akhloufi, T. Celik, and X. Maldague, "Benchmarking of wildland fire colour segmentation algorithms," *IET Image Processing*, vol. 9, no. 12, pp. 1064–1072, 2015.

[28] T. Toulouse, L. Rossi, T. Celik, and M. Akhloufi, "Automatic fire pixel detection using image processing : a comparative analysis of rule-based and machine learning-based methods," *Signal, Image and Video Processing*, 2015.

[29] K. Muhammad, J. Ahmad, S. Member, Z. Lv, and P. Bellavista, "Efficient Deep CNN-Based Fire Detection and Localization in Video Surveillance Applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[30] P. Barmpoutis, K. Dimitropoulos, K. Kaza, and N. Grammalidis, "FIRE DETECTION FROM IMAGES USING FASTER R-CNN AND MULTIDIMENSIONAL TEXTURE ANALYSIS," *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8301–8305, 2019.

[31] Q. Zhang, J. Xu, L. Xu, and H. Guo, "Deep Convolutional Neural Networks for Forest Fire Detection," *Proceedings of the 2016 international forum on management, education and information technology application*, no. Ifmeita, pp. 568–575, 2016.

[32] Q.-x. Zhang, G. Xu, J.-j. Wang, Q.-x. Zhang, G. Xu, J.-j. Wang, Q.-x. Zhang, G.-h. Lin, G. Xu, J.-j. Wang, and J.-j. Wang, "Wildland Smoke Detection Based on Protection Faster R-CNN using Synthetic Smoke Images," *Procedia Engineering*, vol. 211, pp. 441–446, 2017.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[34] J. S. B, O.-c. Granmo, M. Goodwin, and J. T. Fidje, "Deep Convolutional Neural Networks for Fire Detection in Images," *International conference on engineering applications of neural networks*, pp. 183–193, 2017.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2016.

[37] S. Frizzi and R. Kaabi, "Convolutional Neural Network for Video Fire and Smoke Detection," *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pp. 877–882, 2016.

[38] K. Muhammad, S. Khan, M. Elhoseny, S. H. Ahmed, and S. W. Baik, "Efficient Fire Detection for Uncertain Surveillance Environment," *IEEE Transactions on Industrial Informatics*, vol. PP, no. c, p. 1, 2019.

[39] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016.

[40] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - Weakly-supervised learning with convolutional neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. iii, pp. 685–694, 2015.

[41] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with Gaussian edge potentials," *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, pp. 1–9, 2011.

[42] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004.

[43] B. Leibe, J. Matas, N. Sebe, and M. Welling, "Distinct Class-Specific Saliency Maps for Weakly Supervised Semantic Segmentation," pp. VII–IX, 2016.

[44] T. Toulouse, L. Rossi, A. Campana, T. Celik, and M. A. Akhlou, "Computer vision for wild fire research : An evolving image dataset for processing and analysis," *Fire Safety Journal*, vol. 92, pp. 188–194, 2017.

[45] D. Krstinic and T. Jakovcevic, "Image database," Feb 2010. [Online]. Available: http://wildfire.fesb.hr/index.php?option=com_content&view=article&id=49&Itemid=54

[46] Q. Zhang, "Research webpage about smoke detection for fire alarm: Datasets." [Online]. Available: http://smoke.ustc.edu.cn/datasets.htm

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, January 2012, vol. 25.

[48] W. Bae, J. Noh, and G. Kim, "Rethinking class activation mapping for weakly supervised object localization," in *European Conference on Computer Vision*, 2020, pp. 618–634.

[49] G. Perrolas, A. Bernardino, and R. Ribeiro, "Fire and Smoke Detection using CNNs trained with Fully Supervised methods and Search by Quad-Tree," *Proceedings of RECPAD 2020*, pp. 59–60, 2020.

[50] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[51] W. Weng and X. Zhu, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *IEEE Access*, vol. 9, pp. 16 591–16 603, 2021.

[52] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11211 LNCS, pp. 833–851, 2018.

[53] B. Amaral, A. Bernardino, and C. Barata, "Fire and Smoke Detection in Aerial Images," *Proceedings of RECPAD 2020*, pp. 65–66, 2020.