Faculty of Sciences and Technology

Department of Informatics Engineering

Intelligent System for Fire Detection

Ana Abrantes de Abreu Madeira

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Bernardete Ribeiro, co-advised by Prof. Alberto Cardoso and Prof. Catarina Silva and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2020

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Acknowledgements

I would like to express my gratitude to the University of Coimbra and all the people involved in the FireLoc project.

In particular, I would like to thank my advisors, Prof. Bernardete Ribeiro, Prof. Alberto Cardoso, and Prof. Catarina Silva, who contributed throughout the year to guide the experiences and discussions held for the development of the work.

This work was carried out within the scope of the FireLoc project, Ref[a] PCIF/MPG/0128/2017, financed by national funds through FCT - Foundation for Science and Technology, I.P.

FCT Fundação para a Ciência e a Tecnologia

This page is intentionally left blank.

Abstract

The early detection of a fire can largely mitigate its harmful consequences. With the developments in the area of image capture technology and the consequent improvement in image quality, it is now possible to develop systems for visual identification of fire indicators.

The present work presents an intelligent fire and smoke recognition system that can be applied to images captured by smartphone cameras. This system is to be integrated into an application that will allow the reporting of fires using crowdsourced data.

Different deep learning techniques for image classification and object detection were implemented and tested, considering two distinct image object recognition approaches: image classification and object detection. The models' training and evaluation phases are documented in the present thesis as well as all the pre-processing and post-processing steps that were taken into account.

As part of the development of fire detection and classification approaches, different datasets are proposed to train and evaluate ResNet and YOLO models, specific to the fire and smoke recognition problem. The proposed annotated datasets for YOLO models stand out, which can be used in future smoke and fire detection projects.

The proposed system presents promising results for detecting objects of the Fire and Smoke classes in still images. With the proposed detection approach, it is also possible to obtain good results for image classification, assigning a class to each image based on the detected objects with the proposed post-processing method.

Keywords

Deep Learning, Image Recognition, Convolutional Neural Networks, Transfer Learning, Residual Networks, One-stage object detection

This page is intentionally left blank.

Resumo

A detecção de um incêndio na sua fase inicial pode mitigar amplamente as suas consequências. Com os desenvolvimentos na área da tecnologia de captura de imagens e a consequente melhoria da qualidade das imagens obtidas, torna-se hoje em dia possível o desenvolvimento de sistemas de identificação visual de incêndios.

O presente trabalho apresenta um sistema inteligente de reconhecimento de fumo e fogo que pode ser aplicado a imagens capturadas por câmaras de *smartphones*. Este sistema destina-se a ser integrado numa aplicação que permitirá reportar incêndios por meio de dados *crowdsourced*.

No âmbito do desenvolvimento do sistema, diferentes técnicas de *deep learning* para classificação de imagens e detecção de objetos foram implementadas e testadas, considerando duas abordagens distintas de reconhecimento de objetos de imagem: classificação de imagens e detecção de objetos. As fases de treino e avaliação dos modelos são também documentadas no presente trabalho, assim como todas as etapas de pré e pós-processamento consideradas.

Para o desenvolvimento das diferentes abordagens de detecção de objetos e classificação de imagens, são propostos diferentes *datasets* para o treino e avaliação dos modelos *ResNet* e *YOLO*, específicos para o problema de reconhecimento de fumo e fogo em imagens. Destacam-se os *datasets* anotados propostos para treino e teste de modelos *YOLO*, que podem ser usados em futuros projetos de deteção de fumo e fogo.

O sistema proposto apresenta resultados promissores para a detecção de objetos das classes *Fire* e *Smoke* em imagens estáticas. Com a abordagem de detecção proposta, é também possível obter bons resultados na classificação das imagens, atribuindo uma classe a cada imagem com base nos objetos detectados, através do método de pós-processamento proposto.

Palavras-Chave

Deep Learning, Reconhecimento de objetos em imagens, Redes Neuronais Convolucionais, *Transfer Learning*, *Residual Networks*, *One-stage object detection*

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

AP Average Precision. 39, 40, 67

CNN Convolutional Neural Networks. 9, 13–17, 19, 21, 23, 27, 33

COCO Common Objects in Context. 39

FP False Positive. 40

HSV Hue Saturation Value (color model). 8

IoU Intersection over Union. xiii, 18, 19, 39, 40

mAP mean Average Precision. 39, 40, 62, 67

R-CNN Region-based Convolutional Neural Networks. 19

ResNet Residual Network. xiii, xv, 5, 27, 31, 33, 34, 43–48, 50, 53–57, 59, 100

RGB Red Green Blue. xiii, 6, 8, 13, 21, 31, 32

SVMs Support Vector Machines. 8, 9, 19

TP True Positive. 40

YCbCr Luminance; Chroma: Blue; Chroma: Red (color space). 6

YOLO You Only Look Once. xv, 2, 5, 20, 27, 35–39, 60, 65, 73, 100

This page is intentionally left blank.

List of Figures

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

1.1 Context and Objectives

Identifying a forest fire in its early stages is essential for a faster and more effective response from fire and civil protection authorities. An effective initial response can decrease fire damage and, in certain cases, can prevent the loss of lives and property, e.g., houses and other belongings.

Even considering the supplementary preventive measures commonly used, such as the increasing number of firefighting means available, in summer, the number of reported occurrences is alarming [8]. Every summer, there are countless fire occurrences, often caused by humans, and aggravated by the typically high temperatures. The severity of the damage has been particularly visible in recent years. In 2019, Portugal had over 41,000 hectares of land consumed by fire [12]. In the beginning of 2020, wildfires in Australia caused, in less than a week, the destruction of over 1000 homes and a cloud of smoke visible from space [2, 3].

Fire detection is an urge to the population, particularly if followed by a period of communication and report of the current situation to the competent authorities. One of the problems that may affect fire report is the lack of means of locating the person reporting a fire via mobile phone. This situation is particularly worrying when it comes to forest fires where there is no surrounding urban landscape, and it is challenging to use known geographical references. In these cases, someone who wants to report an incident will need to know their exact location to seek assistance. To mitigate this risk, there have recently been more and more initiatives to allow a more accurate location of the call received by the emergency services in Portugal. In 2019, the Government of Portugal announced the implementation of advanced mobile location (AML) technology, which is applied to smartphones, to allow the recognition of calls to the emergency number 112. With this system, it is thus possible to activate geolocation services and automatically send the location coordinates to the 112.pt operational center [1]. However, when locating the person who reports a fire, the location of the ignition may not be immediate, a situation that the FireLoc project aims to resolve.

FireLoc project's [10] primary goal is to develop a system that enables the report of forest fires by identifying, locating, and reporting them, using crowdsourced data. This system includes a mobile application that allows users to send a photograph of the observed fire, as well as geolocation data. Using this system, any user can submit a photograph taken by their smartphone camera. The information sent may also contain the location of the

reported event, allowing a faster and more adequate intervention by the firefighters. Due to the possible increase in the number of contributions, when climacteric conditions are more prone to fire ignition, e.g., during summer, the development of a contribution validation mechanism becomes imperative, allowing for faster and more accurate communication of the occurrence.

With the increase of the volume of information received, processing it by human visual analysis of each reported situation, even if done by an expert, becomes too time-consuming, and therefore not useful for a rapid response. Therefore, the development of an intelligent system that can classify the submitted images automatically is proposed. This system can then be applied to each report made using the FireLoc application, assessing if there is a current fire or not and discarding false reports, which aims to improve firefighters' intervention. This system can enhance the process of analyzing crowdsourced data, ultimately enhancing firefighters' response.

This work is part of the FireLoc project, and its main objective is the development of a system capable of recognizing fire and its signs in the images submitted. The present thesis refers to the validation phase of the submitted images, confirming the presence of smoke or fire. With an intelligent smoke and fire recognition system, it will be possible to make a better management of resources and a better assessment of each contribution in a faster and more efficient way.

The main goals established regarding this work are:

- Study of methodologies applied to image fire recognition;
- Development of a fire recognition algorithm in images;
- Development of an algorithm for processing and classifying the images collected by the application;
- Development of a system for recognizing forest fires in images, to be used to classify crowdsourced images.

1.2 Contributions

While participating in the development of the FireLoc project, with the work performed, the following contributions were made:

- Documentation of the current state of the art regarding fire and smoke detection and classification methodologies;
- Development of a fire and smoke detection system to be integrated in the FireLoc system;
- Proposal of a specific image dataset for forest fire and smoke detection annotated for the training and testing of YOLO networks;
- Testing of the system developed in the project's server using real fire images and analysis of the results obtained;

1.3 Document Outline

This document is organized in a total of five chapters.

This first chapter provides a contextualization of the work developed and main contributions.

Then, in chapter 2, state-of-the-art analysis of smoke and fire recognition in images and videos is presented. Included in this chapter is an overview of current approaches and systems.

Chapter 3 describes the work methodologies and the approach used for the work developed, the proposed methods for recognizing fire and smoke in images, the data used to train the models, and how their performance is evaluated., both approaches proposed.

In chapter 4, the tests performed on the methods used for data pre-processing, training, and performance evaluation of the models are described. The results obtained and their discussion is then presented.

Finally, in chapter 5, a conclusion is given regarding the work developed and the results obtained, as well as a discussion of possible future work to be developed.

This page is intentionally left blank.

Chapter 2

Background and state of the art

In this chapter, the state of the art of fire recognition in images is addressed regarding traditional fire recognition methods and deep learning approaches. First, fire recognition by feature analysis and intelligent fire recognition system approaches are addressed regarding traditional methods.

Then the use of Convolutional Neural Networks such as Residual Network (ResNet) or You Only Look Once (YOLO) networks is addressed, following an analysis of the transfer learning technique.

Current deep learning approaches to similar problems are then critically evaluated and analyzed as to how they can be used in the construction of the fire detection system proposed in this thesis.

2.1 Fire recognition traditional methods

Interest in fire and smoke recognition in images has been increasing and, consequently, new methodologies have been proposed, based on different paradigms.

Such methods are described, distinguishing between methods based on feature analysis and methods based on deep learning approaches. Regarding deep learning, the most relevant architectures and image object detection paradigms are described and reviewed as a possible solution to the fire and smoke recognition problem.

Subsequently, a summary is also given, addressing the different methods that can be applied to video classification with respect to the presence of fire or smoke, as well as a distinction between these methods and the ones used for static image classification.

2.1.1 Fire recognition by feature analysis

The problem to be solved is the recognition of fire in images, referring to the area of image recognition problems. Image object recognition can be defined as the process of identifying and detecting an object or feature in an image.

For this work, object recognition is considered to denote a general problem that includes two distinct tasks, classification, and detection. Object detection refers to the identification and spatial localization of all objects contained in an image, assigning each one a class of a given pre-defined set. On the other hand, the object classification task (or object

categorization [46]) task relates only to the assignment of one or more class labels to an image without the need to obtain the specific location of the objects present [38, 69].

Methods for fire recognition by feature analysis rely on the extraction and identification of the most relevant features for image classification. For each methodology, the extracted features are then used to develop an image or video classification algorithm regarding the presence of fire. These correspond to the initial documented approaches to visual fire recognition both in video and still images. The following methods use image characteristics such as color or texture to identify the presence of fire flames or smoke in static images or videos.

Image features can provide rich information on the image content and can also help identify different regions in an image. Correct discovery and analysis of an image's most relevant features can help solve image recognition tasks.

One common step in feature-based approaches to this problem is the use of color features to isolate image pixels containing the fire. Some solutions have a pre-processing step with the purpose of eliminating the non-relevant information present in the image by filtering it and processing the colors present. The next phase consists on using color-feature extraction methods or texture identification algorithms [28]. This step includes an edge-detecting analysis to isolate the relevant information, making it possible to ignore background objects [21, 68]. One advantage of these methods is that they have a low computing cost.

In statistical color modeling approaches, background objects are extracted, and different sections of the image are segmented into fire and non-fire regions. This segmentation, combined with motion information, is a well-known algorithm for fire detection in video [24]. Most of these approaches only address fire detection, without concern for the presence of smoke. In the early stages of a fire, smoke detection can be of the utmost importance as it allows for faster and more appropriate response by appropriate means such as firefighters.

Using chromatic and dynamic features, fire detection was proposed by Celik et al in [27], using a set of rules. The proposed algorithm thus proved the possibility of image classification based on an analysis of chromatic features to identify pixels containing fire or smoke using the Red Green Blue (RGB) color space [27].

Later on, in 2008, Celik et al proposed a generic color model for fire pixel detection. The algorithm consisted of identifying pixels corresponding to the fire areas of an image. A set of rules was proposed in the Luminance; Chroma: Blue; Chroma: Red (color space) (YCbCr) color space to detect the presence or absence of fire in a certain pixel [25]. Other color spaces can be used for fire detection, with the RGB color space proving to be less robust against illumination change.

Statistics or probability distribution analysis are also proposed as a color-based fire detection method. Typically, this type of method involves the calculation of probabilities of the images containing fire or not. Based on the constructed probability model, it is then possible to classify the images. For image classification, lookup tables or probabilistic models can be used. When using these lookup tables, the goal will be to map the probabilities of belonging to each class and the color distribution present. It is assumed that pixels of the image in which smoke is present are distinguishable from others, forming an isolated set regarding color distribution. For the construction of probabilistic models, it is necessary to calculate the probabilities of the images belonging to each class, based on their color distribution (coordinates in the considered color space). In both approaches, a prior analysis of the histogram of the color levels of the image is necessary to construct the table and calculate the probabilities of belonging to each class [42].

Even though these approaches are computationally efficient, color-based approaches have a high probability of failure with the presence of objects with a color similar to fire, red lighting conditions, presenting a high false-alarm rate [21, 33].

The use of features like color or texture for smoke detection presents additional difficulties. Due to the semi-transparent nature of fire in particular images, color and texture features cannot easily be identified. Smoke can often be mistakenly detected with the presence of clouds or snow, raising false alarms and, therefore, most of the algorithms that rely on color features only focus on fire detection. However, when using video, as there is temporal information, and it is possible to identify the presence of a fire by recognizing scenario changes.

The discovered features can also be used in classic computer vision approaches. Computer vision can be defined as the field that deals with how computers can understand and extract information from images or videos. The use of features for computer vision object detection tasks is, therefore, an essential step in computer vision methodologies. Computer vision algorithms for fire detection usually encompass moving object segmentation, fire pixel detection, and an analysis of the regions containing "fire pixels" detected in the first stages [25]. These approaches will be discussed in the next section, along with other model-based approaches.

2.1.2 Intelligent fire recognition

Following the analysis of fire and smoke traditional recognition methods, a review of intelligent fire recognition approaches is now presented. These are methods that assume the use of a model, such as a neural network, which is trained to recognize flames or smoke in images.

Model-based approaches for fire and smoke recognition can be divided into traditional computer vision methods and deep learning methods. In conventional computer vision approaches, as described earlier, there is a feature extraction phase, previous to the model training phase, that relies on expert knowledge and is the base for the image classification. Deep learning approaches will be addressed in the following sections.

In general, traditional computer vision methods combine the use of feature descriptors with traditional machine learning classification algorithms [48]. Most traditional methods imply the extraction of color, shape, or texture information and, in video recognition approaches, motion information features. Some studies consider the existence of static and dynamic features [53]. The extraction of dynamic features presupposes an analysis of the temporal sequence of events, i.e., the analysis of scenery changes over time. For static features extraction, it is only necessary to analyze static images or, in case of videos, the independent analysis of the frames.

As static features of flames, color, texture, and shape can be useful for identifying fire regions in an image. As dynamic features, in video, changes in the area affected and the edges of the flames, image shape-changing over time and overall movement can also be considered. Changes in the area affected by the fire are identifiable by calculating the number of highlight points, i.e., points where the brightness is above a certain threshold and the overall movement of the flames can be detected based on the video frame sequences [53].

After the extraction of these features, there can be the definition of a rule-model to identify image regions containing the fire, as presented above. Another alternative is the use of

the extracted features for the construction of Feature Vectors, used as input to train the model.

The compilation of the gathered feature information into Feature Vectors is another crucial task that has to be completed before the training phase of the model. These vectors contain information that can help describe the scene present in an image, the present objects, and their most discriminant characteristics.

For these approaches, models such as backpropagation neural networks [53] have also been applied. With the backpropagation algorithm, during the training phase, the neural network's weights are updated according to the total loss and the error that each node is responsible for. By backpropagating the total loss, the nodes responsible for higher error rates are given less importance, i.e., a lower weight value. This algorithm will thus allow the training of a model with good classification accuracy, i.e., the classification of images or videos close to the manually assigned classes.

For the improvement of classification accuracy, some strategies include an initial step of extraction of the color feature parameters to exclude classification interferences based on color [28, 53]. Combining the static and dynamic features calculated, a recent study proposes the construction of multidimensional feature vectors. These are then fed to a backpropagation neural network, allowing the recognition of flame [53].

The use of Support Vector Machines (SVMs) can also be an alternative for fire recognition. SVM is a machine learning algorithm that can be used in classification problems. The algorithm optimizes a hyperplane to separate the data, considering the problem's classes.

One of the proposed algorithms for fire recognition using SVMs begins with the analysis of color distribution values within the range in the image's color model, for example RGB. Following that, using a predefined threshold value, the image's possible fire regions are identified. This algorithm assumes that the surroundings have an orange or red color and that only fires in a more advanced stage present a white-color in their core and uses them to segment the image into fire and non-fire regions [68]. After obtaining the fire images, to discard false candidates, the identified regions pass through a filtering process based on static features, using a trained SVMs. For each defined candidate region, features like color distribution, texture parameter, and shape roundness are extracted and analyzed, discarding possible false positives.

The static features considered include five color distribution features, five texture parameter features, and one shape roundness feature.

As for color distribution features, the ratio of white-yellow pixels, the ratio of red pixels, the ratio of orange pixels, and another two features calculated based on the color histogram of each candidate feature in different color channels.

For the texture features, as in the Hue Saturation Value (color model) (HSV), the H represents color information, the co-occurrence matrix is calculated from that. Texture features such as the angular second moment, the entropy, the mean, the contrast, and the inverse difference moment are then taken into the classification as static features. The co-occurrence matrix is a statistical method used to represent co-occurrent pixel color values. These features are, therefore, helpful in understanding the textures present in the image.

Shape roundness is a boundary analysis feature. It is used as a way to describe the complexity of the shapes present in the candidate region. The use of this feature assumes that more complex shapes will have a higher roundness value. An alternative to the use of this feature, explained under other algorithms presented, would be the calculation of flame

edges, for example.

Combined with dynamic features related to changes throughout the frame sequence, these are used for SVMs training. This makes it possible to identify a fire in video. Examples of the dynamic features used include Variation of contour and Flickering frequency. The calculation of these features uses the coefficients of the discrete Fourier transform in the various frames of the video. By calculating the variance between two consecutive descriptors, changes in the flames over time can be detected. The use of this feature is based on the fact that when watching a video containing fire, there are changes in shape and region of the image representing flame. This distinguishes between fire and fire-colored objects.

For the Flickering frequency calculation, consecutive Fourier descriptors are used over a short time period of the video. Considering these descriptors, the sequence of variances with temporal wavelet is analyzed. This feature allows you to evaluate the behavior of flames over time in the video.

State of the art deep learning approaches imply the use of raw data to train the models and typically use Convolutional Neural Networks (CNN) architectures [48]. These approaches and architectures will be addressed in the following sections.

2.1.3 Fire recognition in video

Traditional approaches to fire recognition in video combine a set of static and dynamic features such as color, shape, texture, and motion orientation for fire recognition [67].

Most video-based fire recognition methods combine the temporal behavior of smoke and flames with its color and shape characteristics to recognize fire. With these features' information, it is possible to use or build a rule-based algorithm or a multi-dimensional feature vector. These feature vectors can then be used as input to a conventional classification algorithm such as Neural Networks or Support Vector Machines [30, 68].

Unlike image analysis, by using video footage, it is possible to obtain useful information on the development of the situation, i.e., the size and evolution of the fire. Adding to the static features that can be extracted from still images, in videos, it is possible to extract dynamic features based on temporal information [53]. It is also easier to identify the location and progression of the event in systems that use static cameras.

There are a few advantages in the use of video for early forest fire detection and report, as opposed to only using still images. While satellite imagery cannot be used for early wildfire detection due to the temporal difference between collection and analysis or the low resolution, solutions based on areal video footage can be seen as a solution [65]. Compared to still images analysis, fire and smoke identification becomes less problematic. However, accurate texture analysis to differ fire from similarly-colored objects in images may be challenging. This problem can be mitigated in video analysis by using features such as edge, shape, or area changing of the identified area as a possible fire [53].

Although the use of video allows for an easier classification, videos usually present a higher data size, i.e., are more complex, and training of these models can be more computationally costly than training for recognition in images. For the current problem, videos recorded with smartphone cameras do not fall under the category of "static cameras", and even more significant difficulties in extracting dynamic features may arise. Besides, detecting changes in scenery over time may not be feasible in shaky, blurred, or poor quality videos, which are very common in videos taken by people in crises. Therefore, this project will focus on analyzing still images.

2.1.4 Current approaches via feature analysis

Given the great advantages of early detection of fire detection, some systems have recently been developed and introduced for this purpose. The following are some of the systems whose algorithms have been previously measured and their current uses in the real world are contextualized.

In the context of RESCUER project development ("Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management") which, similarly to FireLoc, encompasses the development of a system that allows fire reports by crowdsourced data, several algorithms have been developed for fire recognition [15, 20, 28].

By using this system, anyone can send a photograph or video of the incident they are witnessing and reporting any visible fire or smoke in their surroundings. One of the most significant issues was dealing with the high volume of data collected. The number of images and videos submitted became impossible to be visually analyzed by experts in real-time. Therefore, the development of a real-time automatic validation system was imperative to identify which information might be useful, discarding false fire reports [15].

One of the proposed image validation systems was primarily used for fire recognition in social media images. Using this source to form the test dataset, actual system usage conditions were simulated. First, an analysis is made to the images of the different classes to extract features such as color, texture, and shapes present in the images. After calculating these features, Feature Vectors are constructed, aggregating them to be used in the classification of new images [20].

The algorithm used is based on *Instance Based Learning*, and consists of two steps. The first is the use of an evaluation function, which evaluates the proximity between pairs of vectors so that they can be aggregated into sets. Then a classification function is used to receive the vector sets for classification of new images, taking into account the classes considered. A concept description is also maintained to keep track of previous ratings. Fire recognition is then achieved by applying this algorithm to the obtained Feature Vectors [20].

In figure 2.1 is a representation of this algorithm, named *FFireDt*. "The Evaluating Module receives an unlabeled image, represents it executing feature extractor methods and labels it by using the Instance Based Learning Module. The system output (image plus label) interacts with the experts, who may also perform a similarity query" [20]. In this image, *Feature Extraction Method* is referenced as FEM and *Instance-Based Learning* is referenced as IBL.

Another algorithm proposed as a validation method for videos was BowFire and included a frame by frame analysis and, therefore, could also be applied to still images. The still image validation method, BoWFire, consisted of a color-based classification of the photos, keeping only the regions containing "fire" pixels. The result is then combined with another image, product a texture classification, only including the "fire-texture" pixels. If these images are coincident, the image is classified as fire [28].

The application of traditional pattern recognition approaches to fire and smoke recognition is one of the methods that has been widely explored. However, these imply the additional step of identification and extraction of most discriminating features, which can be difficult for this problem [50].

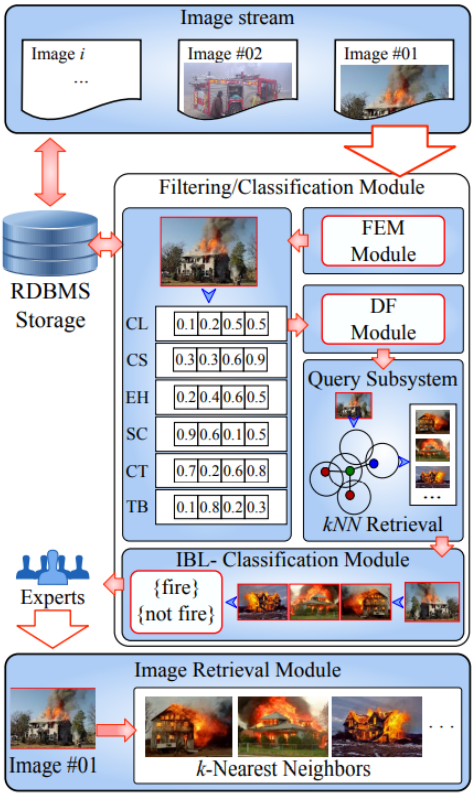


Figure 2.1: "Architecture of the FFireDt" [20]

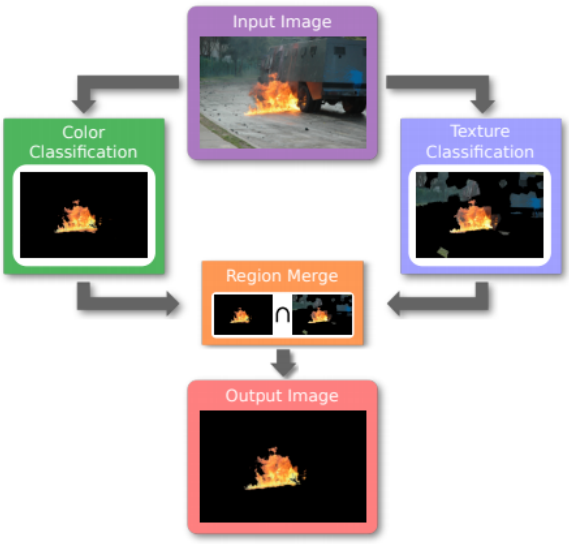


Figure 2.2: "Architecture of the BoWFire method" [28]

2.2 Deep Learning approaches

Following the analysis of approaches for intelligent fire and smoke recognition in Section 2.1.2, state of the art computer vision approaches and deep learning approaches will now be analyzed.

In traditional or classic computer vision approaches it is possible to identify several classic phases of image classification. Figure 2.3 shows a step-wise division of the image analysis and classification process when using a classification model.

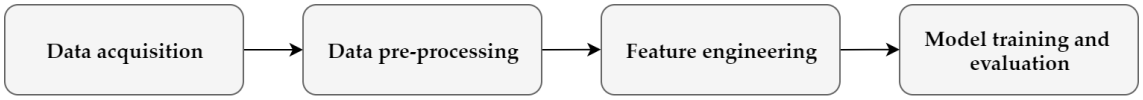


Figure 2.3: Phases of image classification

Data acquisition:

This step involves the acquisition of images to be used in training and testing of the model as well as all the dataset preparation necessary. Each image needs to be analyzed and annotated according to its category.

There should be a large enough volume of data for the model to learn. For fire detection, the set of images should contain a large enough number of examples of images containing fire, images containing smoke and images containing forest areas, considered neutral.

The acquired data should also be representative of all possible situations. It should include images containing clouds as not smoke or containing sunsets as not fire, as they may be confused with the presence of smoke or fire.

Data pre-processing:

An additional pre-processing step may be necessary before the images are given as input to the model. This step ensures that the input provided for training is in the expected format and ready to be used in the next stage.

A good pre-processing of the data can make model training easier and allow better classification accuracy.

Feature engineering:

In this stage, all the relevant features that can be used for classification are obtained from the training set of images. In Traditional Computer Vision approaches, feature extraction includes manual extraction and selection of features.

Model training and evaluation:

With the features extracted in the previous stage, the model is then trained, iterating over the available data. After training the model, metrics like a confusion matrix or accuracy can be used to evaluate its performance.

2.2.1 Convolutional Neural Networks

As an alternative to classic computer vision algorithms for image classification, state of the art algorithms use deep learning methods. The idea is to use deeper architectures, with the ability to learn more complex features, which will result in better performance in object

recognition problems in images [43]. One of the advantages of using deep learning methods is that feature extraction is done directly by the model, allowing end-to-end learning, i.e., the model only needs a set of annotated images as an input [48].

With the use of deep learning methods, it is possible to process large amounts of information to train very deep architectures with multiple layers. The extraction of discriminative features is done automatically. By processing large amounts of data for training, it becomes possible to obtain better generalization ability and, therefore, higher accuracy in the testing phase [50].

By using this type of methods, the patterns present in the data are discovered automatically, and the most descriptive features identified in each class are then used for classification.

The high capacity to learn complex characteristics in images that these models have, combined with proper training algorithms, allows the understanding of the information present in images with multiple levels of abstraction [43].

Convolutional Neural Networks (CNNs or ConvNets) are the most successful type of models used for image classification and object detection tasks [48, 67] and also the most representative model of deep learning [43].

Since these networks are able to automatically learn a set of visual features from the training data, the results don't rely solely on expert knowledge to build relevant feature extractors. It is, therefore, possible to obtain more accurate classifications [30].

Several approaches to the problem include the study and optimization of a model built and trained from scratch. However, the full training of a model requires the availability of large amounts of annotated data for training, validation, and testing [62]. The training of deep neural networks can take a very long time. However, an additional data pre-processing stage can make use of traditional computer vision techniques to facilitate the training of the model or to make it more robust [48].

As stated before, it is possible to use a CNN operating directly on raw data. Some state-of-the-art proposed algorithms for fire and smoke detection assume the training of the model with a raw RGB frames as input and no previous feature extraction phase [30, 67].

When using CNNs, there are some essential concepts to have in mind:

- **Filters :**

  Filters are used as feature extractors and are used to identify different characteristics of the images such as edges, vertical or horizontal lines, allowing the perception of the outline of objects present, for example. Therefore, filters are applied (through convolution operations) to the image, generating different feature maps. Therefore, by applying a filter in different convolutional areas, it becomes possible to extract the various features of an image [50].

- **Local Perception :**

  To identify the objects present in the image, it is necessary to capture the Local Dependencies in the images. CNN follows the principle that there is no need to have a global perception of the image by all neurons of the neural network. Each neural node can, therefore, focus on one part of the image and then integrate the information with the others [50].

- **Sub-sampling :**

Allows the reduction of computational complexity. By applying downsampling, it is possible to identify features that are invariant to translation [50]. Sub-sampling is performed by the network's pooling layers. There are usually several convolution and pooling layers in CNNs.

- **Full connection :**

  At the end of the network, the most correlated features are selected from all the abstract features extracted. The fully connected layer takes as inputs the features extracted by the previous convolution and pooling layers and predicts the correct label, classifying the image.

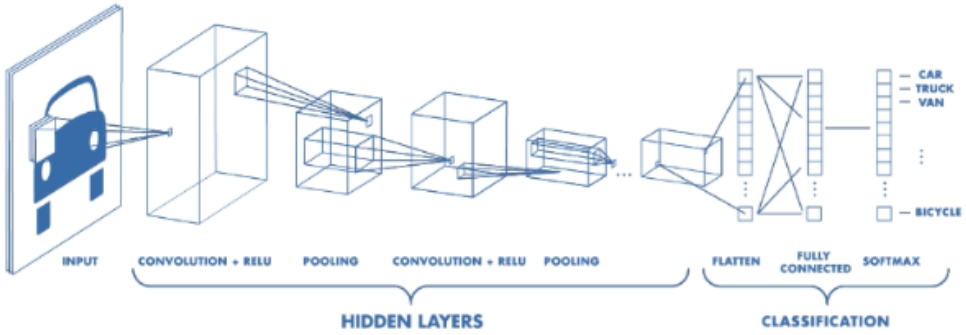


Figure 2.4: An example representation of CNN layers, from [13]

Figure 2.4 shows an example of a CNN architecture.

Convolution Neural Networks, or CNNs, are multi-layered feedforward deep learning neural networks, with different types of layers, widely used for image analysis. The main advantages of CNNs are the ability to learn representations with a great capacity for abstraction. This allows for solving complex problems like image object recognition [46].

The name of these networks is derived from the convolution blocks. A convolution operation can be defined as a mathematical operation that expresses what two different sets of information have in common. Each convolution operation, in CNNs, can be seen as filters that, during training, "slide over the input", generating a feature map. The output of the convolutional layers then passes through an activation function. The resulting feature maps, after all the convolution operations, are put together, resulting in the output of the convolutional layer [13].

Max and average pooling are also important layers of a CNN. These are responsible for subsampling operations. In these layers, the input is divided into rectangular pooling regions. Then, max-pooling layers compute the maximum of each region, and average-pooling layers compute the average of each region [7].

Following the network's pooling and convolution layers, there usually is a fully connected layer. All of the neurons that compose a fully connected layer are connected to all the neurons in the previous layer. When solving classification problems, the number of neurons in the last fully connected layer corresponds to the number of classes considered [7].

When choosing a model there are some very distinct architectures that can be considered. Depending on the problem that needs to be solved different paradigms can affect the results obtained, even if using the same training (and validation) data.

Training complex architectures can be a difficult task. Due to the complexity of the networks and the vanishing gradients problem, training a CNN can take a long time and demand a lot of computational capacity [62]. As such, it is essential to choose the architecture to use appropriate to the problem that is to be solved. In the following sections, some state-of-the-art models for image object recognition are presented.

2.2.2 Residual Neural Networks (ResNets)

One common strategy to enhance the network's performance is to increase its depth, however, the deeper the network, the harder it is to train efficiently. With increased depth, optimization becomes more difficult and there is an increased probability of overfitting [34].

Overfitting happens when a function models a particular set of data too well, i.e., in a model, when the training set's noise or random fluctuations are learned by the model and used as a future reference for performing classification. The problem with overfitting is that these characteristics may be particular to the training set and not applicable to all possible cases, leading to a more significant classification error.

Activation functions are a significant part of a neural network's architecture, conditioning its performance and its ability to learn. These functions define the output of the nodes when given a particular input. They are what allow neural networks to comprehend and learn from data. When using deep learning, the effect of adding more layers and using certain activation functions can cause the gradient (error) to increase exponentially, resulting in a very slow training of the network's front layers. Another problem is that CNNs take a long time to train, due to having more training parameters [62].

As a solution to prevent overfitting, Microsoft has made advances in terms of optimization by using residual learning networks, ResNets. Unlike with other CNN architectures, with these networks, it becomes possible to use deeper networks without compromising test accuracy [34].

Degradations in training deep neural networks are a known problem [59]. As the depth increases, there is a point where the training accuracy starts decreasing.

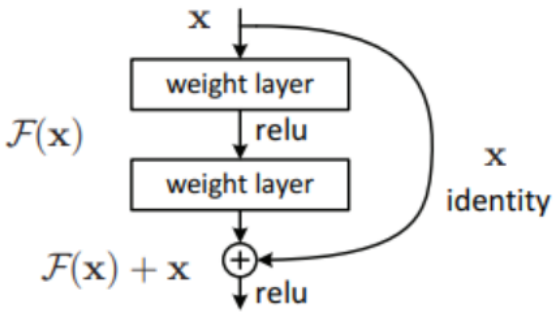


Figure 2.5: "Residual learning: a building block" [34]

In ResNet architectures, instead of only using convolutions, the degradation problem is addressed by using "shortcut connections" or identity mappings to increase network performance, represented in 2.5. The use of shortcuts as identity mapping consists of using residual learning "every few stacked layers" [34]. After the training of each block of layers, the rectified linear units (ReLU) activation function is used instead of other less sensitive functions. By using the ReLU function after each block of layers, it is possible to evaluate if the input contains useful information and, therefore, decrease the error in comparison to other plain networks. In this case, when the result is negative after the training of the block, it is possible to consider it 0, avoiding the error propagation [34][51]. This also helps prevent the deterioration of the model accuracy. It also allows a faster and easier training phase [62].

Another common approach to enhance the model's performance is making the training set closer to the situation it will be used in. As the model will be used to identify forest fires, images containing urban settings and other backgrounds may be confusing and not as useful for training.

Another possibility is to use an unbalanced dataset. Some models trained for similar applications, i.e. identification of fire or smoke in images, use an unbalanced dataset for the training of the model. They propose that a higher number of neutral images compared to the number of fire images used can help decrease the probability of false-positive occurrence.

In the same study, it is also proposed that very deep convolutional models for fire recognition tend to overfit very easily. As a solution to that, they propose their own architecture, creating their own CNN architecture and training it from scratch [37]. Although they were able to obtain a good result, to train this network, a considerable amount of data was needed. Training also took a very long time, due to the number of parameters that needed to be optimized and the complex processing operations required.

Since its publication, in 2015, ResNet models have been applied in image classification and object recognition problems, achieving very promising results.

There are several versions of the ResNet, defined by the numbers 18, 34, 50, 101, and 152, with the numbers corresponding to the number of network layers. ResNet 18 and 34 use building blocks similar to the one shown in figure 2.5, two layers deep. The remaining ResNets 50,101 and 152 are made up of 3-layer building blocks.

The most important feature of these architectures, which distinguishes them from all others, is precisely the use of the "shortcut connections" mentioned above, at the end of each of these blocks. Making use of these properties, deep residual learning networks are the ones that can present greater depth without reflecting on their performance, among all the examples of architectures presented.

Examples of state-of-the-art architectures for CNN are, for example, the above mentioned residual networks or ResNets, Inception, Mobilenet, and VGG neural networks. Each of them has some singularities that set them apart from other previously used architectures. These are explained below to clarify the advantages and disadvantages of using each of them.

VGG architecture

VGG models were first proposed in 2014 by the Visual Geometry Group (VGG) from the University of Oxford. These architectures stand out from previous ones, mainly due to the concept they use to increase their depth [61].

For the construction of these models, it is assumed that by combining several smaller filters, it is possible to obtain results similar to those resulting from the use of larger filters. Therefore, very small convolutional filters are used as their depth is increased. By maintaining the benefits of using small filters, it is possible to decrease the number of parameters that need to be optimized throughout the workout. In these networks, the ReLU activation function mentioned above is used [61].

There are several versions of the VGG architecture, VGG16, and VGG19. While the VGG16 architecture has only 13 convolutional layers, VGG19 has a higher depth with 16 convolutional layers. Although these networks get consistent results in terms of image classification, they use a considerable disk size and take a long time to train.

MobileNet

MobileNet's architecture was proposed in 2017 by Google, suitable for use contexts where there is no large computing power available. They stand out from the rest because, although they are lightweight models and occupy low disk space, they achieve good accuracy in image classification problems [36].

There are two available versions of MobileNets, called MobileNet V1 and MobileNets V2. From the first to the second version, some improvements have been implemented with respect to its performance, thus being better suited for systems where memory access limitations exist.

Using these algorithms consider each image as whole, identifying Fire or Smoke becomes possible by classifying it as Neutral, Fire or Smoke. However, when it comes to images of forest fires, especially in their initial phase, fire and smoke may be superimposed, making the classification task difficult.

2.2.3 You Only Look Once (YOLO)

As an alternative to the aforementioned image classification algorithms, recent research has shown the efficacy of considering as part of the object detection problem, the localization as well as the classification of existing objects in an image [26, 45, 69]. Therefore, the object detection problem is defined as the identification of where an object is located and which class it belongs to in a given image. Object detection allows for a complete understanding of the images by not only solving the classification task but also by estimating the object's region in the images [32, 69].

As such, there has been significant progress in the development of CNN models for detecting objects in images, with the development of several frameworks, organized into two distinct categories:

- **Two stage detection frameworks**, in which the process is divided into two different phases: The first one can be considered a pre-processing step, in which candidate regions that may contain objects are proposed. And a second phase that consists of excluding false positives and classifying each of the objects found in the image.
- **One stage (Unified) detection frameworks**, which perform the process at once, without the initial region proposal step.

The functioning of this type of frameworks assumes the understanding of some fundamental concepts:

- **Bounding Boxes**

  The examples of frameworks presented use boxes designed to delimit the objects present in the image to identify the location of the detected objects. These boxes are defined as bounding boxes.

- **Anchor Boxes**

  Anchor boxes or anchors are not a common concept to all frameworks presented. To make training more efficient, some algorithms propose fixing the initial boxes' dimensions to be used to identify the location of objects. This initial set of boxes with fixed dimensions is defined as anchors.

  These are defined for training models and can be adapted to training data using dimension clusters. The anchors' adaptation of dimensions assumes that objects' sizes in training data are similar to those of the objects present in the images used for testing the models. This process can improve the accuracy of the detections.

- **Intersection over Union (IoU)**

  For the evaluation of the detections obtained by the models, the value of the overlap area over the joining area between each box identified by the network and each box of the annotation with which there is overlap is calculated. For a better visualization of this evaluation measure see figure 2.6

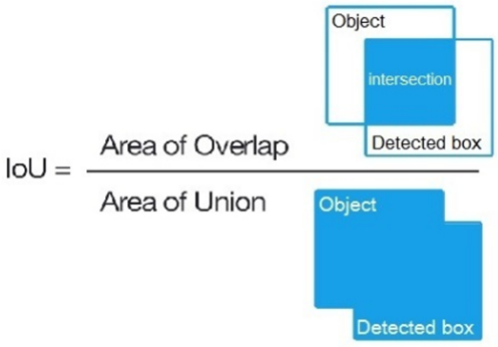


Figure 2.6: Intersection over Union (IoU) visualization

After assigning a certain value between 0 and 1 to the IoU threshold, it is possible to consider:

1. **True Positive (TP)**: An object is detected correctly if the IoU value obtained is equal or higher than the defined threshold;
2. **False Positive (FP)**: An object is detected incorrectly if the IoU value obtained is lower than the defined threshold;
3. **False Negative (FN)**: The object is considered not to have been detected is there is no corresponding box in the annotation. In other words, if there is an occurrence of smoke or fire identified in the test dataset and not detected by the network, it is considered a False Negative;
4. In the results obtained with these models, it is not possible to identify True Negatives, since the boxes only indicate the presence of fire or smoke.

- **Localization Errors**

  The detection frameworks presented have as main objective the localization of objects in the image space, using bounding boxes. Therefore, it is considered that when a predicted box does not overlap sufficiently with the box defined in the image annotation, a localization error occurs. It is also considered that a localization error occurs when there are multiple detections of the same object [35, 46].

  This type of error is one of the most frequent causes of the occurrence of false positives in the detections, which impairs the model's evaluation. Consequently, to avoid this type of error, it is important to define an IoU threshold for assessing detections appropriate to the problem domain.

Two stage detection frameworks

One of the first two stage object detection frameworks that introduced the concept of bounding boxes for the identification of regions in images with the location of the present objects was the Region-based Convolutional Neural Networks (R-CNN) [32, 46].

Initially proposed in 2013, R-CNN architectures are suitable for image object detection problems and can, therefore, be adapted for image fire recognition problems. The solution presented by these networks to the problem can be divided into two main phases: region proposal and classification [32].

In the first region proposal stage, candidate bounding boxes are obtained in the image that may contain objects from any class, through selective search [40, 46]. These boxes independent of the object's aspect ratio or size of the objects [32].

In the second stage, the classification is made based on the first result, fine-tuning the regions, discarding false positives. Each candidate box's features are extracted and used to build feature vectors. These vectors are then used to classify the present objects in the image, using a set of class-specific SVMs [32, 40].

Due to the high computational cost and slowness of R-CNN training, changes have been proposed to reduce the training time of this network, such as Fast R-CNN [31] and Faster R-CNN [58]. Contrary to what happened with R-CNN, in Fast R-CNN networks, features are extracted from the entire image before identifying candidate regions. This difference made it possible to do the training process end-to-end [31, 38].

In the Faster R-CNN networks, the fundamental concept of anchor boxes was introduced. These anchors consist of multi-scale boxes that are used as a reference for the delimitation of objects in images. The pre-definition of reference sizes for bounding boxes simplifies the region proposal process, making the network faster than the previous ones [38, 58].

Faster R-CNN presents significant improvements in image processing speed and the accuracy of classification over the previous models. However, in all of these frameworks, the training comprises a multistage pipeline, with a lengthy region proposal step that slows down the analysis of each image [46], which can be a problem when the model's response time is relevant.

One stage detection frameworks

Unified or one-stage detectors perform object detection using the entire image. This type of framework allowed the extraction of features from an image. These are then used to directly predict the bounding boxes that contain the objects present, as well as the probabilities of these belonging to the classes considered [38]. Therefore, the detection of objects is made from the entirety of an image and using a single CNN network, so there is no need for the

region proposal step [46].

One of the first one-stage detector frameworks was the YOLO, proposed in 2016 [55]. The analysis of the image in its entirety allows for the network to obtain better information of the context, making the occurrence of false positives less likely. However, due to the use of initial fixed anchor boxes, compared to Fast R-CNN, this network showed a greater tendency for localization errors in the detections performed.

Its successors, YOLOv2 or YOLO9000 [56], have already addressed the issue of anchor box sizes. If the size of the object found in the image is not close to any anchor box, the network may not be able to identify it. To avoid this, YOLOv2 proposes adapting the anchor boxes' dimensions using k-means and training using various scales. Because these pipelines rely on skipping the region proposal stage, the disparity of sizes between the objects present in the images can make their detection difficult [46].

To mitigate this problem, newer versions like YOLOv3 [57] or YOLOv4 [22] analyze each image's features using different resolutions. As versions later than YOLOv2 maintain the anchor box adaptation mechanism specific to the dataset, it helps in detecting objects of various scales, thus reducing the number of false positives [46]. Moreover, unlike their predecessors, these networks use multi-label classification, which allows correct detection in scenarios with overlapping objects of different types [22, 57].

These more recent versions of YOLO analyze the image at three different scales, obtaining several feature maps to detect each object localization. These models can detect small objects with greater precision. However, YOLOv3 has more difficulty with medium-sized objects [57].

In these networks, it is also possible to define the size of the grid division that is considered in the analysis of the input images. Increasing or decreasing the resolution at which each image is analyzed can help detect objects with very large or small dimensions.

2.2.4 Transfer learning

As stated in the previous section, one of the disadvantages of training a deep model is the amount of data necessary to obtain good results. The most common strategies used to reduce annotation efforts are dataset augmentation and transfer learning.

The use of synthetic data to increase the training dataset has been a strategy used to combat the lack of annotated datasets in object recognition problems [54]. Some studies propose the use of synthetic data to train the model for the problem of smoke recognition in images, that is, the artificial addition of smoke in neutral images [67].

Other strategies for dataset augmentation are the use of horizontal or vertical image shifts, cropping or scaling the images, adding them to the original data to increase the available dataset [62]. This can not only be useful as a way to compensate for the unavailability of training data but can also boost model performance, making it more robust [62].

When the annotated data available for training isn't enough to obtain good results, it may be necessary to use a pre-trained model optimized to solve a different problem and fine-tune it. A known technique is the adaptation of a model previously trained for another domain, Transfer Learning.

The use of transfer learning encompasses a few issues. When using this technique, discovering what knowledge to transfer is an important task, i.e., knowing what part of the information should be preserved from the previous training. If done correctly, it will allow

the knowledge information transfer to be useful for the current problem's domain classification [52]. When using transfer learning, usually, the "classification" (see figure 2.4) part of the CNN is re-trained for adaptation to the new problem domain. Within this part of the network, the issue is choosing what layers to retrain.

It is also necessary to know what pre-trained model to use and what layers to retrain in order to adapt it to the new problem's domain [52]. This method can be applied to fire recognition problems by using the pre-trained weights of a network for another target domain and adapting them for feature extraction taking into account the classes of our problem, which facilitates the learning of the common features in both domains [54].

The most common strategy is to fine-tune the last layers of the network, training them with the new dataset, and adapting them to the intended problem's domain. The weights of the first layers are maintained, responsible for the extraction of more low-level features, and the last layers are retrained, extracting more complex features, specific to the problem's domain [45, 46]. Transfer learning can thus make the model training process faster and improve the models' ability to learn [44].

State-of-the-art models, as the ones mentioned in the previous section, can be used by applying pre-trained weights available, resulting from training models with any of the above-defined architectures to solve other classification problems or object recognition in images. With the initialization on the network using some pre-trained weights and using transfer learning techniques it is then possible to adapt to the fire and smoke image classification problem's domain.

It is also possible to combine both strategies, using synthetic data or other strategies for dataset augmentation to fine-tune a pre-trained model, applying transfer learning [54, 65].

2.2.5 Current deep learning approaches

This section presents examples of intelligent systems for recognizing fire in images.

In this study, [50], the CNN network takes as input videos or images and gives as output the probabilities of it belonging to each class considered. A model is created with a custom architecture for the problem, called the *MCCNN model*. All videos are first converted to images frames and all images are then resized to fit the input size of the model. Each image is given to the model as a matrix with each RGB channel, and discriminating features are extracted. Parameter adjustment was achieved using backpropagation. The resulting model could be applied to both video and image inputs, classifying them as belonging to one of the classes considered, **Fire** or **Not Fire** [50]. In figure 2.7, from [50], is presented by the authors as a visual explanation of the algorithm used for data pre-processing, training, and classification using the model.

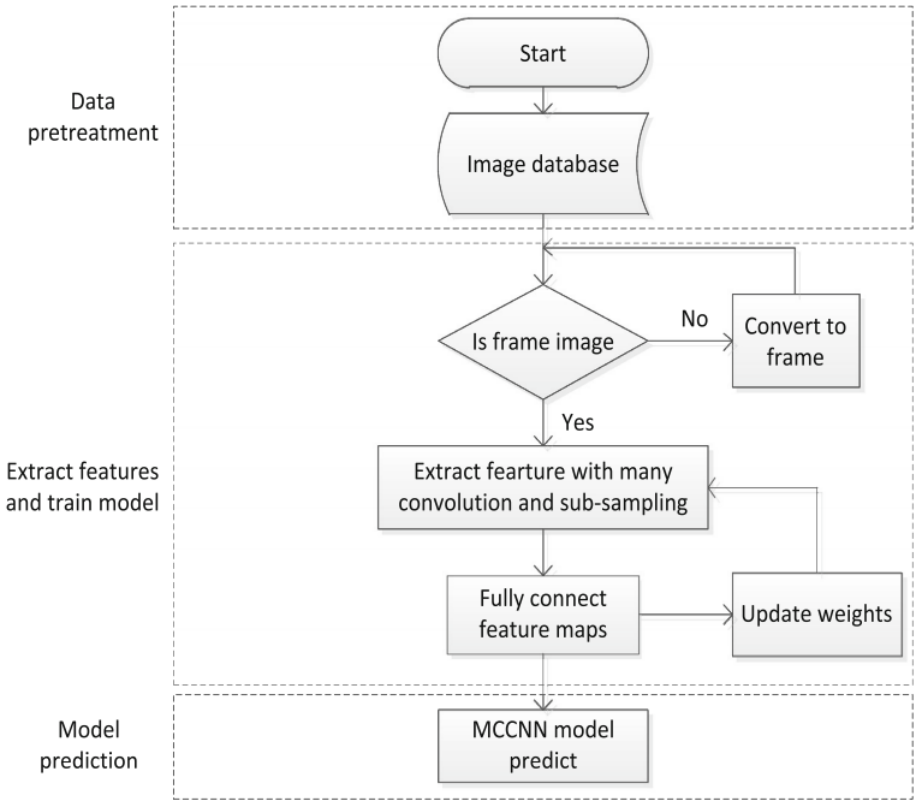


Figure 2.7: "Flowchart of fire recognition" [50]

Faster R-CNNs have also been successfully applied for smoke detection tasks. The principle behind the use of these networks is its success and accuracy in object detection tasks [58]. This study proposes the use of a Faster R-CNN to identify smoke regions, thus recognizing smoke in videos [67].

One of the advantages of using these types of models is the possibility of identifying the location, in the image, where the fire is recognized. In figure 2.8 is a diagram of the algorithm used for smoke detection using a Faster R-CNN model, from [67].

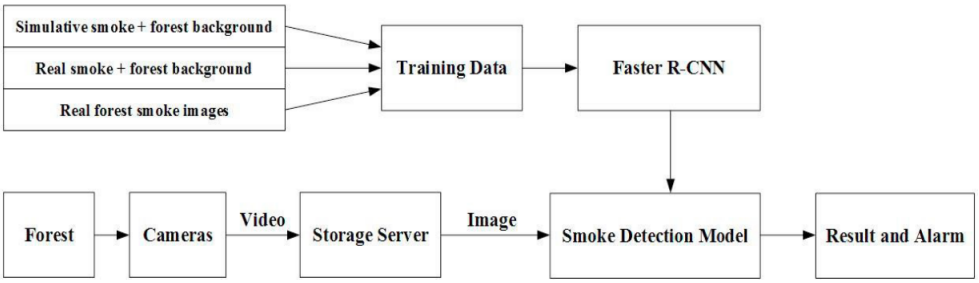


Figure 2.8: "Flowchart of forest smoke detection" [67]

As a way to circumvent the lack of data recorded for training the model, the strategy of artificial insertion of smoke in images with forest background was adopted. The procedure for artificial insertion of smoke consists of the use of smoke simulation software, to create smoke columns in forest images and the inclusion of real images of smoke columns in a forest background, adjusting brightness and smoke concentration in the final image, making it more realistic. Authentic forest smoke images were also used to enrich the training dataset. Using the dataset created, it was then possible to train the Fast R-CNN model.

Another recent system used "captioning and classification of dangerous situations" to autonomously detect anomalies, for robot applications. The adopted approach makes use of a CNN to identify anomalous situations in images. For this system's classification task, the Inception model was used, which allowed the construction of a trainable end-to-end architecture. In this system, the Inception [64] module was used for recognition of other situations: "broken windows, injures people, fights, car accidents, guns and domestic violence" [19].

2.3 Conclusion

Given the state-of-the-art fire recognition systems, some relevant considerations are pointed below.

There is a great deal of interest today in developing alternatives for the automatic visual recognition of fire and smoke. However, many of these are not specific for the recognition of forest fires, and some of them just consider classifying images as either fire or non-fire-containing.

There is also a lack of available datasets for training ResNet or networks, annotated according to the *Fire*, *Smoke*, and *Neutral* classes, and a lack of YOLO annotated datasets, with the fire and smoke objects marked in the images. In addition, many of the available datasets do not contain specific forest fire circumstances, including urban situations with objects such as cars and houses, which can be confusing to detect.

Systems based on feature analysis are simpler and, in general, computationally less demanding compared to other approaches presented. However, they show weaknesses in situations of changing usage scenarios, which can be unreliable when it comes to false positives. Most of these systems have as their main component color analysis, which implies that the system has little reliability in unfavorable lighting conditions, such as fog. This type of approach still has little capacity for generalization since the classification accuracy can change significantly when applied in other contexts.

Intelligent fire or smoke detection systems are more complex and therefore, in general, computationally more demanding. However, in general, they have a better generalization capacity, since the learning carried out by the models is independent of the context of use.

The use of deep learning makes it possible to extract features from images automatically. The main disadvantage is the need to have a greater number of examples so that there is sufficient training of a model to avoid overfitting. One of the main difficulties is, therefore, the lack of annotated datasets.

As ways to overcome this problem, several methods of generating images have been proposed for training the models, applying transformations to images without changing their classification (such as rotation or selective cropping of the images).

It is also possible to use transfer learning, which consists of using the weights of pre-trained models to detect objects in the domain of other problems, taking advantage of their ability to identify the contours of objects and shapes present in an image. The use of these methods has positive effects on training the model, making it more robust.

In conclusion, based on the state-of-the-art analysis of fire and smoke recognition in images, two image recognition approaches will be studied and evaluated: image classification and object detection. While in the image classification approach, the images are classified

as a whole, in the image object detection approach, which follows a different paradigm, the smoke and fire present are detected and located in the image. This means that the smoke columns resulting from a fire and existing fire spots are detected and marked with a bounding box in each image's space when using the object detection approach.

In the following chapter, both approaches considered are explained regarding the steps followed concerning the training and testing phases of the models.

This page is intentionally left blank.

Chapter 3

Fire Recognition Approaches

This chapter presents the work performed to study both fire recognition approaches proposed. The image classification and object detection approaches are presented and characterized as to the datasets used to train the models, the studied pre-processing methods, and the evaluation metrics used to assess the performance of the models obtained.

As part of the FireLoc project, this work has as its primary objective the development of an intelligent fire and smoke recognition system in still images taken using smartphone cameras. For this system, fire recognition is performed, taking into account two different approaches: image classification and object detection.

By discarding false reports, which will be identified as neutral images, and determining which reports are sightings of flames or smoke in forest territory, it is possible to have better incident reporting. Therefore, the developed system will help in the decision of intervention to be taken by the fire fighting means.

The first part of development focused on studying previous approaches to similar problems, such as object detection, fire and smoke recognition in images, as well as previously existing systems used for identification and reporting of fires in both urban and forest contexts.

In the development of the intelligent fire recognition system, the implemented methodologies are based on Deep Learning approaches, focusing on state-of-the-art models for image object recognition: Convolutional Neural Networks (CNNs). Using these types of models makes it possible to automatically extract the most relevant features observed in the images and, consequently, obtain better classification results.

Other techniques, such as transfer learning and dataset augmentation, were also considered for the algorithm's development. Moreover, methods for improving existing datasets and developing the specific dataset were also explored.

The developed system also goes through a testing phase, using photographs taken in the context in which it will be used (forest fire conditions), to evaluate its performance. For the system to be reliable, a testing phase is required to show that no significant errors are observed. This means that for the system to be used in real conditions, it is ideal to obtain a high true-positive rate.

The lack of a benchmark dataset for fire recognition led to a specific dataset proposal for the problem. For the generation of the proposed dataset, the need for it to be representative of all possible situations and provide a sufficient number of examples from each class to allow model training and evaluation to be possible was taken into account.

3.1 Fire Recognition System Setup

This section presents the image classification and image object detection approaches regarding the models, datasets used, and pre-processing methods. The metrics for the evaluation of the models' performance used for each approach are also presented.

Due to a lack of available datasets, specific to forest fire, and smoke recognition, the Transfer Learning technique was used. This was done by initializing the models with pre-trained weights optimized for recognizing the *ImageNet* dataset [11]. For the image classification approach for the development of the system, the Residual Network (ResNet) [34] models were used. As for the object detection approached, the models used were YOLOv3 [57] and v4 [22].

The development strategy consisted, in a first instance, of a model training phase, in which the networks were initialized with the weights corresponding to the optimization for the *ImageNet* object recognition problem and fine-tuned to adapt to the new domain considered. After training the models, the next phase was the evaluation of their test results.

Given the context in which this system is intended to be applied, there will be images captured with different smartphones. This means that there is no uniform resolution or coloration of submitted photographs. In addition, the photos will be captured by people present of the ground, which can also result in image imperfections, such as blurred images or strange lighting conditions that can hinder classification performance.

In order to facilitate feature capture by the network, data preparation or pre-processing was needed. Therefore, different formats, sizes, and qualities of images used in the training and performance evaluation of the model were also considered. The model names and datasets used for the image classification and object detection approaches can be observed in table 3.1. In the following sections, the datasets mentioned are explained as well as the differences between the variations of the ResNet and You Only Look Once (YOLO) models used.

Considered Approach	**Trained Models**	**Training and Testing Datasets**
Image Classification	ResNet 18, ResNet 34, ResNet 50, ResNet 101, ResNet 152	Fire-Smoke-Cropped, Real-Images-Cropped
Image Object Detection	YOLOv3, YOLOv4	Fire-Smoke-YOLO, Real-Images-YOLO

Table 3.1: Models and datasets used in both approaches: image classification and object detetion

3.2 Image Classification

For the image classification approach, the ResNet models were studied and applied to this problem. A study of the different possibilities for image pre-processing was carried out as a step for them to be used as input for these models. Only with the pre-processing phase completed, these images were used in the training and performance evaluation phases of the models.

Considering that ResNets are Convolutional Neural Networks (CNN) models, they are able to extract useful features from images and generally achieve high accuracy in object classification and detection tasks. In addition, due to the existence of "shortcut connections"

for every n blocks in their architecture, the performance of these models does not degrade as significantly with the increase in the depth of the network. The tests carried out in the first semester aimed to study the feasibility of using these models in the fire recognition system.

In this approach, three different classes are considered for image classification: *Fire*, *Neutral*, and *Smoke*.

The Neutral class includes images in which there is no smoke or fire flames present. To the Smoke class belong all the images where there is smoke presence, having no visible flame. The Fire class includes images with fire flames present.

A distinction is made between images containing fire and images containing only smoke since the presence of flames is more indicative of the presence of a fire. While in a picture with flames, it is safe to say that there is a fire that can be used to infer the location of the occurrence, in images where only smoke is visible, it is not possible to assertively locate the fire that gave rise to it.

To adapt the pre-trained model to the new problem domain, in order to apply the transfer learning technique, it is necessary to change ResNet's last fully connected layer, from the 1000 classes considered for the *Imagenet* dataset, to the three classes considered in this problem. This network layer is then re-trained, adapting the classification to the domain currently considered for the *Fire*, *Smoke*, and *Neutral* image classes. In other words, each test required the adaptation of the models' output layer to size three (which is the number of classes considered) and then fine-tuning the models by re-training this layer exclusively.

In figure 3.1 is a diagram representing the method of the image classification system in both the training and testing phases.

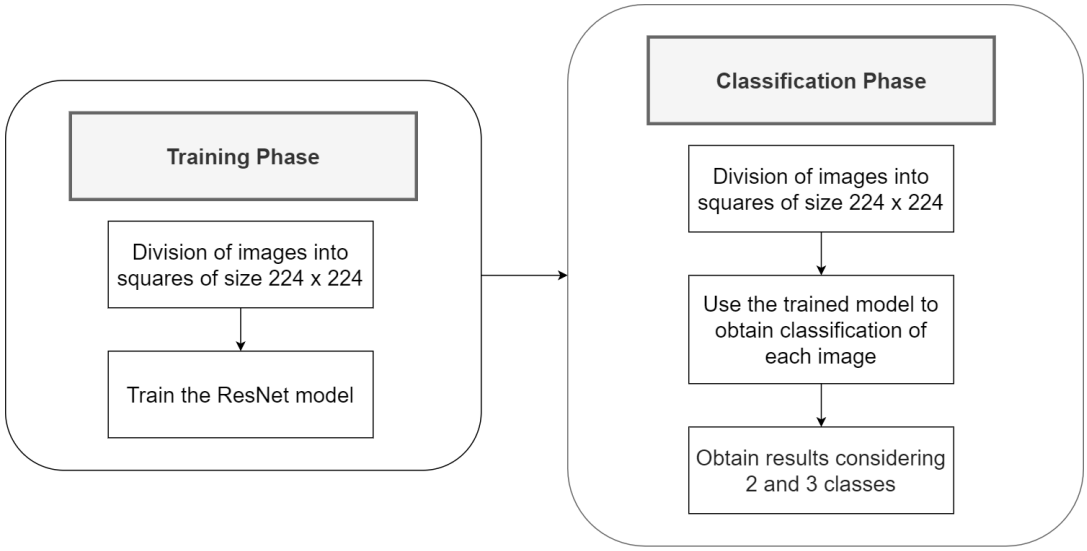


Figure 3.1: Image Classification Approach

As described in figure 3.1, the classification approach consists of a training phase followed by a test phase of the models.

The models' training phase consists of using the images divided into parts of 224 by 224, followed by the process of fine-tuning the models, after adapting the architecture to the classification problem using three classes. To re-train the models, the parameters to be used must be defined in advance. These were optimized in the tests performed in order to improve the models' ability to classify images.

In the testing phase, parts of images of size 224 by 224 are also used. Re-trained models are then used to classify the images. The classifications obtained are performed considering the **Fire**, **Smoke** and **Neutral** classes for each image. The results then go through a post-processing process, in which only the **Fire or Smoke** and **Neutral** classes are considered.

3.2.1 Image Classification Datasets

Open Source Image Data For the training of these models an open-source dataset was used, from [9]. This dataset is balanced, i.e., contains an equal amount of images from each category (*Smoke*, *Fire*, and *Neutral*), with 1000 images per category for training and validation. The images are split into train and validation sets: 70 % are used for training and 30 % for validation of the models. This dataset is named *Fire-Smoke-Dataset*, and some examples of the set of images can be seen in 3.2 figure, with an indicative grid of their actual size.

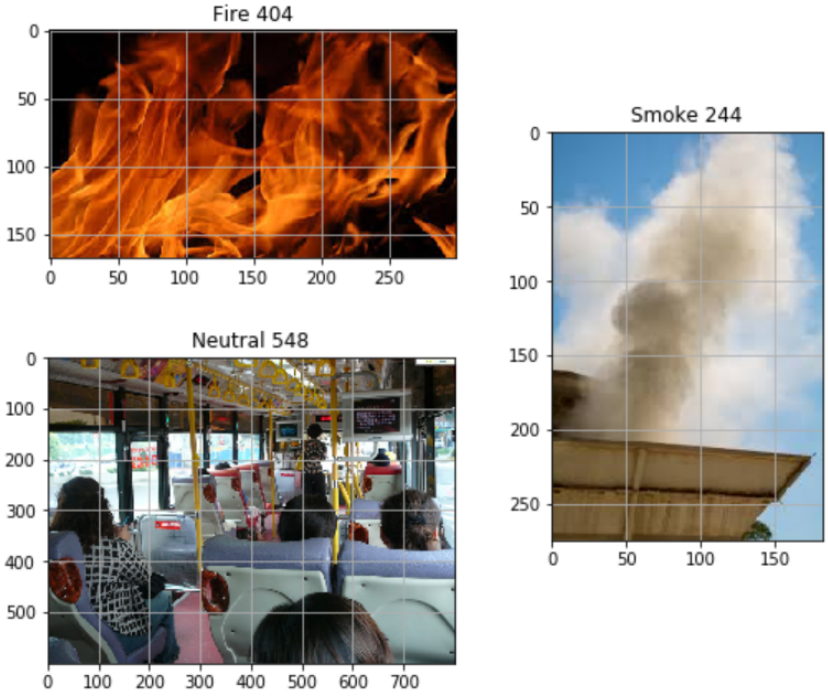


Figure 3.2: Images from the *Fire-Smoke-Dataset*

Image Data from Simulacrum [1]

Another set of photographs was provided, taken in a simulacrum, performed by firefighters. The pictures were taken using different smartphones and tablets, thus allowing them to simulate real conditions of use of the system. These images all correspond to the context of forest fires. A representation of all the classification classes was also taken into consideration, meaning that in this specific image dataset, there are images with smoke, others with fire, and others without any evidence of fire.

The dataset of specific images contains a total of 429 images, 167 identified as Fire, 74 identified as Neural, and 188 identified as smoke. These sets of images can be considered a

[1]Images collected on May 15, 2019, in tests carried out in Serra da Lousã by *Associação para o Desenvolvimento e Aerodinâmica Industrial (ADAI)*

representation of real-world situations in which the system will be used, with varying sizes and proportions being taken in different orientations, i.e., horizontal and others vertical. Some of the present images have characteristics such as blurred parts, the presence of objects in the background (such as fire engines or wind blades), or lighting conditions that can be confused with the presence of smoke, which can make the classification task more difficult. These characteristics bring images closer to those that may be submitted by users of the application and bring this dataset closer to what are expected to be the real conditions of use. This dataset is named *Real-Images-Dataset*, and some examples of the set of images can be seen in figure 3.3, with a grid indicating their actual size.

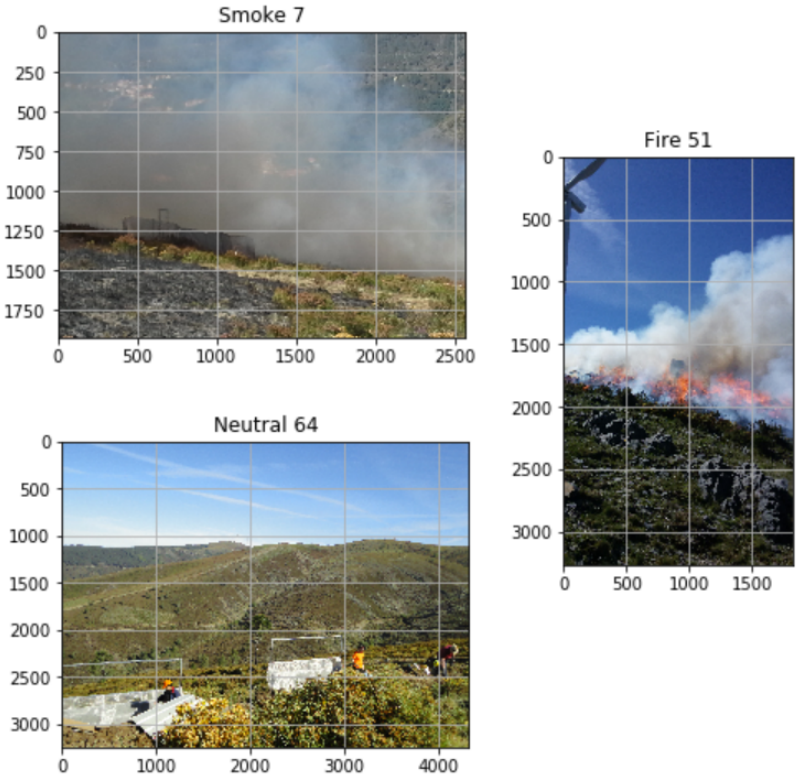


Figure 3.3: Images from the *Real-Images-Dataset*

There are noticeable differences between both datasets in terms of size, presence of objects, and situations represented. In the first dataset, there are not only images corresponding to forest fires but also indoor ignitions and smoke accumulations. As an opposite to that, the photos taken in a fire drill context only consider wildland ignitions with fewer people and objects, different lighting conditions, clouds, and were all taken in a non-urban environment. However, due to the low amount of images available corresponding to each class, in this approach, the *Real-Images-Dataset* was only used for model testing and performance evaluation.

There is also a considerable difference between the datasets' image sizes, which can cause additional difficulties in classification. For better visualization of the size discrepancy, see table 3.2.

	Max shapes			Min shapes		
Dataset	**Fire class**	**Smoke class**	**Neutral class**	**Fire class**	**Smoke class**	**Neutral class**
Fire-Smoke	300, 429	307, 427	2304, 3072	108, 168	118, 96	145, 176
Real-Images	4320, 4320	4320, 4320	3240, 4320	1836, 1836	1836, 1836	1836, 2560

Table 3.2: Dimension of datasets' images

3.2.2 Data pre-processing

ResNet's model training using transfer learning technique, requires an initial image pre-processing step to facilitate feature extraction. The weights with which the networks are initialized correspond to the optimization result for object recognition in the *ImageNet* dataset, from the "ImageNet Large Scale Visual Recognition Challenge" [60].

The transfer learning strategy used is to freeze the weights of the first layers of the network, responsible for extracting more low-level features, and fine-tune the latter, adapting the network's ability to identify the characteristics present in the image to the domain of the fire and smoke detection problem. As such, feature extraction from input images follows a series of conventions, which must be kept for the images given to the network.

These conventions include adapting each image to 224 by 224 size and splitting the image into Red Green Blue (RGB) channels. The pre-training parameters of the networks must be maintained, as in [41], so that it is possible, during training, to use the weights assigned to the first layers to recognize shapes and forms in the images of the new dataset, specific to the problem.

For training and testing ResNet models, all images from the training and test datasets were split into squares of size 224 x 224, creating the *Fire-Smoke-Cropped*. Examples of images of each class can be seen in figure 3.4.

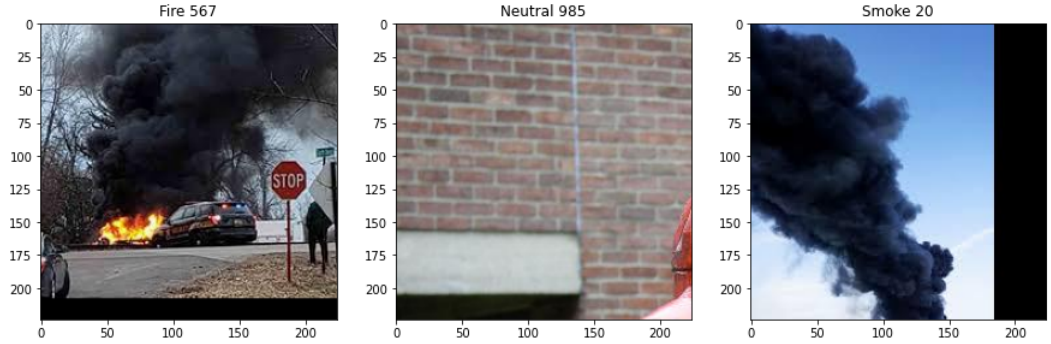


Figure 3.4: Images from the *Fire-Smoke-Cropped* dataset

For the new images to remain class-persistent, it was necessary to manually review the annotation of the training and validation images, as well as manual annotation of the test images. Due to the large size of the test images, 100 images (of size 224 x 224) of each class were randomly selected and manually annotated. These images make the *Real-Images-Cropped* dataset, and examples from each class can be seen in figure 3.5.

As mentioned before, the classification approach included a separation into the different RGB channels, as exemplified in figure 3.6.

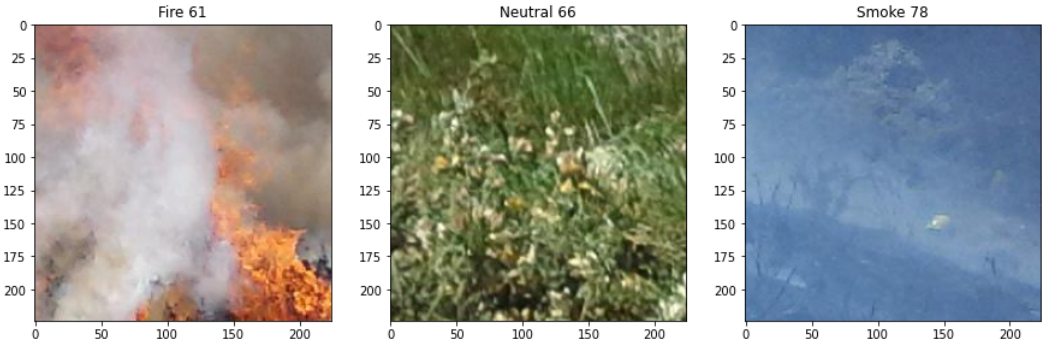


Figure 3.5: Images from the *Real-Images-Cropped* dataset

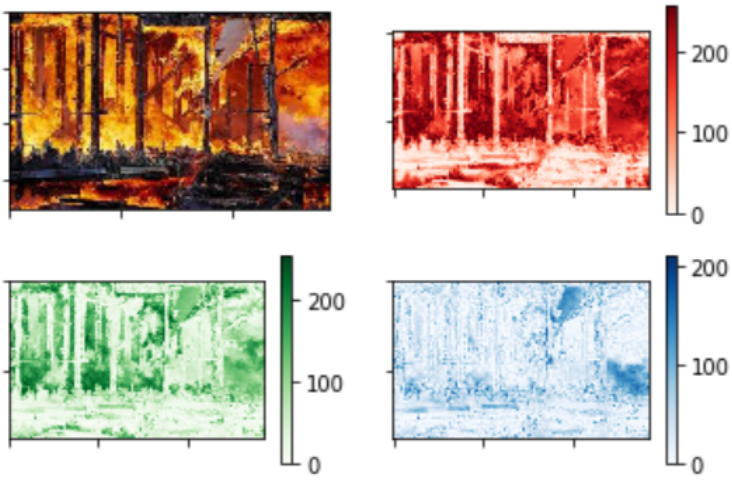


Figure 3.6: RGB components of an image

The image was also normalized to fit values within the range [-1, 1]. Thus it became possible to mitigate possible detrimental effects of the noise present in the images in the identification of features. This step can also facilitate the extraction of relevant features by standardizing the information on each image before training the neural network.

The images in the training dataset are also shuffled. This means that, at every epoch of training of the model, the images will be organized in a different order. Therefore, reshuffling the data at each iteration can contribute to better training of the model and help avoid early overfit due to possible patterns in class order.

Only the fully connected layer of the model is retrained. For that, the dataset used is the *Fire-Smoke-Dataset*. After the pre-processing steps necessary for the use of transfer-learning or specific to each test are performed, the model iterates over the 900 images of each class in the training set. A validation step with the other 100 images is done by the end of each iteration.

After training the models, these are tested with images from the *Real-Images-Dataset*, in order to understand their behavior in real conditions of use.

In the training of neural networks, among the optimized parameters (such as the weights assigned to each node), the parameters whose values are chosen to control the learning process can be defined as hyper-parameters [66]. The optimization of hyper-parameters can help CNN training, improving the performance of models (such as learning rate). However, when adjusting the learning rate, errors may arise detrimental to the model's training [63].

The Stochastic Gradient Descent (SGD) with momentum algorithm [63] emerged, using momentum-based acceleration, that can speed up the network's training without deteriorating the results.

As mentioned above, for the application of transfer learning, the ResNet models were initialized with the training weights for the *Imagenet* dataset, and the pre-trained models available in [16] were used. As a simplification adopted, the initial tests were carried out using ResNet 50.

The initial tests consisted of adapting the learning rate and batch size parameters to be used in the training of the networks. For these tests, the ResNet 50 network was used, with the pre-training parameters fixed, varying only the parameter to be optimized.

3.2.3 Evaluation of the models

To evaluate the performance of the trained models, the following metrics were used: training and validation learning curves, confusion matrix and F1-score.

The **F1-score** metric can be interpreted as a weighted accuracy value, regardless of the number of images belonging to each class of the dataset.

First, Precision and Recall are calculated for each class. Precision is the number of true positives of each class divided by the total number of images to which it is assigned by the model.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

Recall (true positive rate or sensitivity) is the number of true positives of each class divided by the number of ground-truths images belonging to that class.

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

The F1-score metric used to evaluate the ResNet model is the weighted average between precision and recall. It takes into account false positive and false negative predictions made by the model.

$$\text{F1-score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.3}$$

This measure, adapted from [4], is calculated in a weighted way, taking into account the classification of all images, regardless of their ground-truth class. However, the F1-score values obtained for each class can be useful to discover flaws in the classification, for example, where there is more confusion between classes.

After training, the **confusion matrix** obtained in the test set, adapted from [6], was also analyzed. The observation of the obtained confusion matrices allows for the verification, in greater detail, of the model performance in the classification of the test data. It can also be useful to understand the most significant classification errors and improve the model's training, as it shows the comparison between the labels of the images predicted by the model and the true labels.

Choosing the analysis of these matrices without normalizing the values allows dealing with absolute numbers and letting us know exactly how many images were classified correctly or incorrectly. Therefore, the objective will be to obtain the largest possible number of images whose labels in the annotation coincide with the classification made by the model, observable in the diagonal of the matrix.

In addition to evaluating the model's test performance, the **training and validation learning curves** are observed, which represent the evolution of the training and validation loss respectively.

This step helps to understand the evolution of the models' performance throughout the training epochs and to adjust them to obtain the greatest possible accuracy. Additionally, observing the learning curve's graph makes it possible to detect errors that may occur during training.

In this graph, a decrease in loss values obtained by the model in the training and validation curves should be observed, with the increase in the number of training epochs. It is possible, when comparing both learning curves, to evaluate the generalization ability of the models at the end of training and identify possible underfitting or overfitting situations.

Both F1-score and confusion matrix metrics were analyzed taking into account three classes of **Fire**, **Smoke** and **Neutral** and two classes of **Fire or Smoke** and **Neutral**. As a post-processing step of the test results, the **Fire** and **Smoke** classes are merged, and the confusion matrix and F1-score are recalculated. The new analysis allows the evaluation of the network's performance, excluding the confusion between fire and smoke that can be seen in images in which both are present.

3.3 Object Detection

For the object detection approach, the YOLO models were studied and applied to this problem. Considering the detection of each object in the image space, a more apparent distinction between fire and smoke becomes possible. However, due to the frequent overlap of objects of both classes, only YOLO models from version 3 or higher were considered. The implementation of these frameworks, available in [18], was used as the basis, corresponding to the published works [57] and [22].

As previously mentioned, these models take the images as their input, without the need of a fixed size, and propose bounding boxes corresponding to the objects found for each one. Therefore, another advantage of using the object detection approach will then be the possibility to analyze images of all possible sizes, without any additional pre-processing steps.

Figure 3.7 is a diagram representing the method of the image classification system in both the training and testing phases.

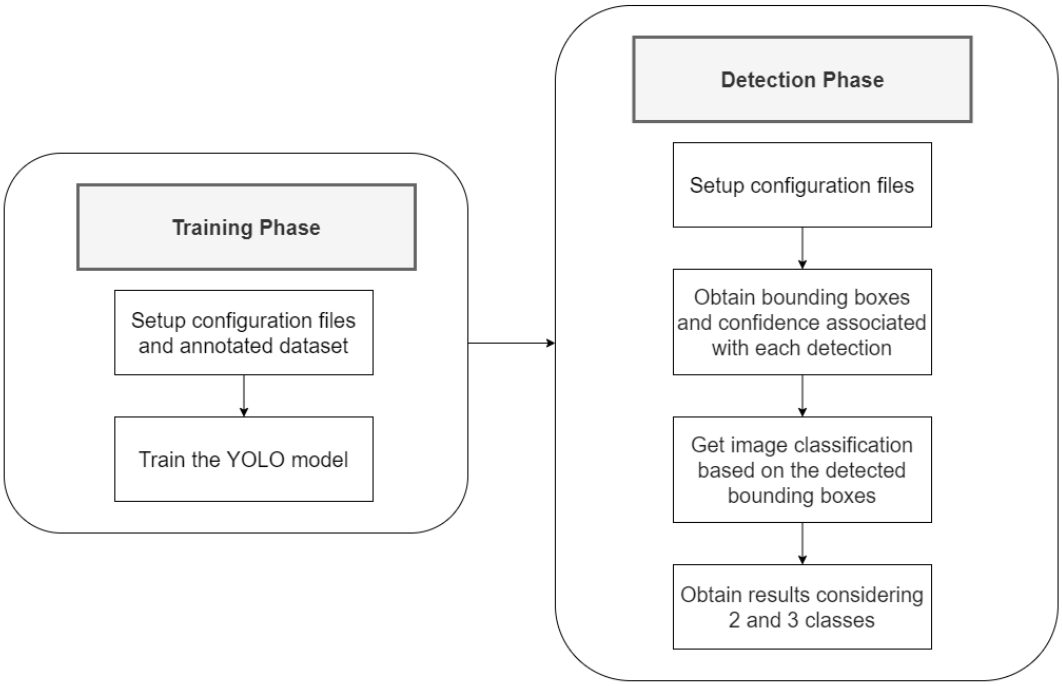


Figure 3.7: Image Object Detection Approach

As described in figure 3.7, the image object detection approach consists of a training phase followed by a test phase of the YOLO models.

In the training phase, images annotated according to the YOLO format are used to train the models. First, the training parameters and pre-processing parameters are defined in the configuration files. These were optimized in the tests performed in order to improve the models' ability to detect the correct location of fire and smoke in images. The weights resulting from the pre-training with the *Imagenet* dataset are used for initializing the YOLO models, which are then fine-tuned with the annotated dataset images.

In the testing phase, images annotated according to the YOLO format are also used. The re-trained models and the respective configuration files, obtained in the previous phase, are

then used to detect fire and smoke in the images. The models give as output for each image, the bounding boxes that identify the detected objects, and the confidence associated with each box. The results then go through a post-processing process, in which, based on the boxes detected in each image, classes Fire, Smoke, or Neutral are assigned to each image. Similar to the classification approach, these can be analyzed considering the classes Fire Smoke and Neutral or Fire or Smoke and Neutral.

3.3.1 Object Detection Datasets

For the training of YOLO models, there needs to be a specific annotation of each image and, typically, even using the transfer learning method, a large number of images is necessary to obtain good classification results. In order to make it possible to use the same datasets, they were annotated manually, delimiting all parts of the image where the presence of smoke or fire is observable with bounding boxes.

In this approach, two different object classes are considered: **Fire**, and **Smoke**. The images considered by the classification approach presented in the previous section as belonging to the **Neutral** class are those in which no object is detected.

For these networks' training, the annotation of the images must be carried out differently, marking, in each image, all existing objects of each class with a bounding box. The images need to be annotated according to the YOLO annotation format, as explained in [17].

The images annotated according to the YOLO format can be used to train and test all YOLO models and, follow the following guidelines:

- The object classes considered were [0, 1] are [Fire, Smoke];
- For each image there is a txt file with the same name as the annotated image;
- Each file line defines one bounding box location;
- Each bounding box contains one object present in the image;
- Each line of the file, to define a bounding box, must contain <class> <x> <y> <width> <height>;
- The values <class> <x> <y> <width> <height> are relative to the image's dimensions, between [0.0, 1.0], where:

$$< x >= \frac{<\text{x object position}>}{<\text{image width}>} \tag{3.4}$$

- The coordinates [x, y] correspond to the center of the bounding box and the position [0, 0] corresponds to the upper left corner of the image;

For example, an image that shows smoke throughout the space will be annotated:

1 0.5 0.5 1.0 1.0

For this purpose, a graphical image annotation tool was used, *LabelImg* [14]. It allows viewing the image, drawing the boxes delimiting the objects present, and generates, for each image, a txt file with the annotation in YOLO format.

In the annotated images, each column of smoke is marked with a **Smoke** class bounding box, and each fire spot is marked with a **Fire** class bounding box. When drawing each

bounding box, the objective will be to delimit only one object in each box and avoid including much of the background, in order to prevent false positives and localization errors.

The datasets used for training and testing the ResNets networks in the previous approach were annotated according to the YOLO annotation format, creating the new *Fire-Smoke-YOLO* and *Real-Images-YOLO* datasets. The annotated images were then used for training and testing the YOLOv3 and YOLOv4 networks.

Examples of annotated images from the training dataset, *Fire-Smole-YOLO*, can be seen in images 3.8, 3.9 and 3.10. These examples were obtained as a result of annotating the *Fire-Smoke-Dataset* dataset, using the *LabelImg* tool.

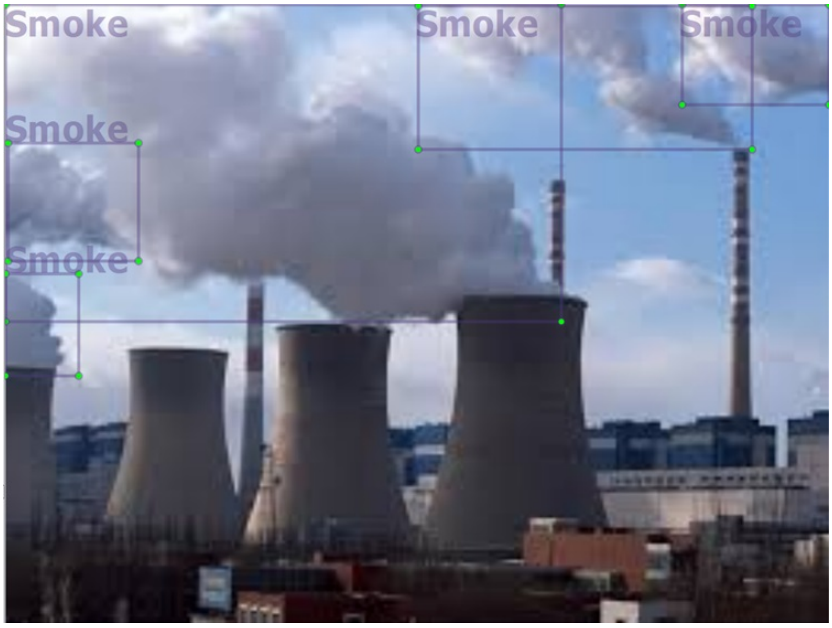


Figure 3.8: Example of image with smoke, from *Fire-Smoke-YOLO* dataset

Figure 3.9: Example of image with fire, from *Fire-Smoke-YOLO* dataset

Figure 3.8 is an image annotation an image that only contains examples of smoke and figure 3.9 is an example of an image annotation from the training dataset that only includes examples of fire. Image 3.10 is an example where objects belonging to both considered classes are present.

Figure 3.10: Example of image with fire and smoke, from *Fire-Smoke-YOLO* dataset

By observing these examples of images, it becomes apparent that the advantage of using the object detection approach is the possibility of differentiating between fire and smoke in the same image, unlike the classification approach.

3.3.2 Data pre-processing

After manually annotating the images according to the YOLO format and the implicit generation of txt files for all images, they were used to train the networks.

The fact that the annotation uses relative coordinates, and the analysis of each image is made according to different resolutions allows images of different dimensions to be used without distortion or the need for additional pre-processing. Another advantage of using YOLO neural networks over the use of ResNets is the location of objects in the image space using bounding boxes. The separation of fire and smoke in the same image allows for less ambiguous object detection and can avoid classification errors.

To train the YOLOv3 and YOLOv4 models, the annotation txt files and the images to which they correspond must be in the same location, and a *train.txt* file is required, describing the name and location of each image to be used for training.

It is also necessary to create an *obj.data* file in which the number of classes to be considered and the file that identifies the images to be used for training the model, *train.txt*, are defined.

An *obj.names* file must also be generated. This file contains the names of the classes to be considered. The classes must be in the correct order, one per each line of the file, corresponding to the number considered in the annotation. In an annotation in which **Fire** and **Smoke** are considered classes 0 and 1 respectively, in the *obj.names* file, **Fire** should appear on the first file line, and **Smoke** should appear on the second line.

Several optional steps were taken into account for the tests performed, to improve the classification performance of YOLO networks. One of them is the adaptation of the anchors' size to the training dataset, which consists of the analysis of the training images, annotated according to the same principle of the test images, and generation of the initial anchors' sizes using k-means, as proposed in [57].

For training models with the *Fire-Smoke-YOLO* dataset, using the object detection ap-

proach, it was necessary to define:

- The network initialization with weights of a model pre-trained with the *Imagenet* dataset, available in the repository [18];
- Batch size and subdivision values, which were defined according to memory limitations, following the recommendations from YOLOv3 [57] and YOLOv4 [22] publications;
- The number of classes, which in the fire and smoke image detection problem is two: Fire and Smoke;
- And the initial size of the bounding boxes (or anchors) used by the network to mark the objects found, the anchors. The boxes with which the objects identified by the network are marked on each image depending on the initial anchors defined and, as such, it is one of the parameters to be optimized [57].

3.3.3 Evaluation of the models

For the evaluation of these models, using the Intersection over Union (IoU) calculation, the following metrics were used: precision-recall curve; true positive, false positive and false negative count; mean Average Precision (mAP) and Average Precision (AP) values per class; and confusion matrix.

As stated above, IoU is the measure of accuracy for each detection in YOLO models. This measure is calculated between each object detected by the model and its corresponding box in the test images. To determine if a predicted box is assigned correctly, a threshold needs to be defined. If the IoU calculated between the predicted box and the box in the annotation is 1, it means there is a perfect match between the two, reaching the maximum overlap possible.

For the evaluation of YOLOv3 and YOLOv4 models' performance, the IoU threshold was set at 0.3, lower than the typical 0.5 or 0.75 values used for Common Objects in Context (COCO) dataset detection [5]. This decision was based on the fact that, to the human eye, it becomes difficult to distinguish between correct predictions considering a threshold of 0.5 and 0.3, according to [22, 57]. When considering a lower IoU threshold, more inaccuracies in the detected objects' location will be tolerated, and, as such, it will be possible to view a more significant number of valid detections (avoiding false negatives in the analysis of each image).

The mAP value analysis allows the evaluation of the performance of the model in terms of the detections made. It is a standard measure of the performance of these models, and for this work, an adaptation of [29] was used, from the project [23].

One of the reasons for the analysis of mAP is that, by considering the level of confidence in the detections made, it is possible to evaluate the relationship between false positives and false negatives [29].

Precision can be defined as the correction of model detections. A high precision value means that most of the object detections were done correctly. Recall refers to the object's ability to detect objects in the model. A high recall value indicates that most of the objects in the images have been detected [49]. The goal is to obtain a model that can identify most of the objects present in the images, avoiding the possible increase of false positives.

In order to calculate the mAP value, the following steps are performed:

- First, the models' detections are ordered in descending order of the confidence value and are matched to the ground-truth objects in the test dataset, considering an IoU threshold of 0.3. For penalization of multiple detections of the same object, as in [29], a match is only considered a true positive if the ground truth bounding box hasn't been previously matched to another detection.

  For each match made, the precision and recall values are then calculated.

- After being matched, the detections and corresponding ground truths are then used to calculate the precision-recall curve. Precision-recall pairs are calculated for each set of detection - ground-truth boxes, considering the maximum precision found for any recall level (interpolated precision) [49].

- The precision-recall curve is constructed by calculating the accumulated Precision and Recall values of the detections made (True Positive (TP) or False Positive (FP)), varying the confidence threshold in the detection. The graph has as (x, y) axes the precision-recall pairs calculated in the previous step.

- The AP value for each class approximates the area below this curve and, consequently, the mAP value will result from the average of the obtained AP values.

As a post-processing step, the images were also evaluated as a whole. For this, they were given a classification considering the classes of the previous approach: **Fire**, **Smoke**, and **Neutral**. Therefore the classes assigned were:

- If a bounding box is assigned to the image, detecting a *fire* object, the class **Fire** is assigned to the image;

- If there are assigned no boxes containing as *Fire* objects and there are *smoke* objects detected, class **Smoke** is assigned to the image;

- Only in cases where no objects are detected in the image, the image is assigned the **Neutral** class.

Similar to the previous approach, the merge of **Fire** and **Smoke** classes into **Fire or Smoke** and **Neutral** classes was also considered.

For the analysis of the results by image, after the post-processing step described above, confusion matrices related to the test results were generated and analyzed.

The matrices were generated considering three and two classes (in the original form and after the merge). In conjunction with the graphical observation of the detections, that is, the boxes drawn in the image space, the confusion matrices' analysis is useful to assess the performance of the models not only in the detection but above all in the classification task.

The analysis of these matrices was also useful for adjusting the confidence threshold to be considered. The definition of this value is essential for the proper functioning of the model since detections whose confidence indicated by the model are below the confidence threshold are excluded. If the value is set too low, there will be a higher number of erroneous detections, and otherwise, with too high a threshold, insufficient objects may be detected. The analysis of the classification results by image allows for the assessment of the effects of the variation of this value in the detection of fire and smoke in all images.

Although the classification approach implies the use of simpler models, that is, less deep ones that imply shorter training times, the distinction between fire and smoke in the images

becomes difficult since, both being present in the same image, the Fire classes and Smoke can easily be confused.

In order to evaluate both approaches presented, several tests were performed. The following chapter presents the tests performed and the results obtained for each of the approaches presented.

This page is intentionally left blank.

Chapter 4

Results and discussion

This chapter presents the different tests carried out concerning the two approaches: image classification and object detection. The tests performed regarding the pre-processing of the images for training the models and optimizing some training and test parameters are presented. The results obtained are then presented and discussed, using the metrics defined in the previous chapter.

4.1 Results obtained with ResNets

This section presents the tests carried out to develop the image classification approach. Different tests were performed with the Residual Network (ResNet) models taking into account the pre-processing of the images and the optimization of some training parameters of the model.

Initial tests

The initial tests were aimed to choose the model to be used to optimize training parameters (such as batch size and learning rate) in the following tests. As such, the variations in the architecture of the simplest ResNets (that is, with less depth) with two and three layers between each "skip connection" were considered: ResNet 18 and 50.

The tests had the same setup as used in the pre-training of these models and consisted of training up to 200 epochs and evaluating the F1-score and confusion matrix test results. The setup details, as described in [41], are:

- Training and validation with *Fire-Smoke-Cropped* dataset:
  - Training images consisted of 70% of the 1000 images in each class (2100 images), of size 224 x 224;
  - Validation images consisted of the remaining 30% of the 1000 training images each class (900 images), also of size 224x224.
- Fixed batch size and learning rate values, as used in the pre-training of the networks: batch size 32, learning rate 0.001;
- Use of ResNet model variations: ResNet 18 and ResNet 50 pre-trained on the *Imagenet* dataset;
- Training up to 200 epochs;

- Tests with images from the *Real-Images-Cropped* dataset: 100 images of each class (300 images) of size 224 x 224.

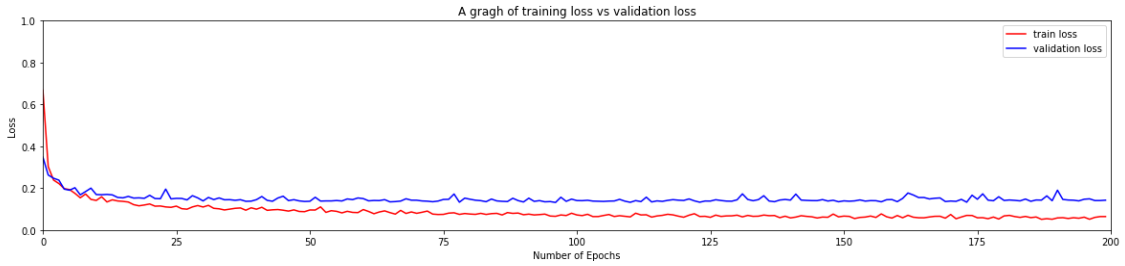


Figure 4.1: Training and validation learning curves from the training of ResNet 18

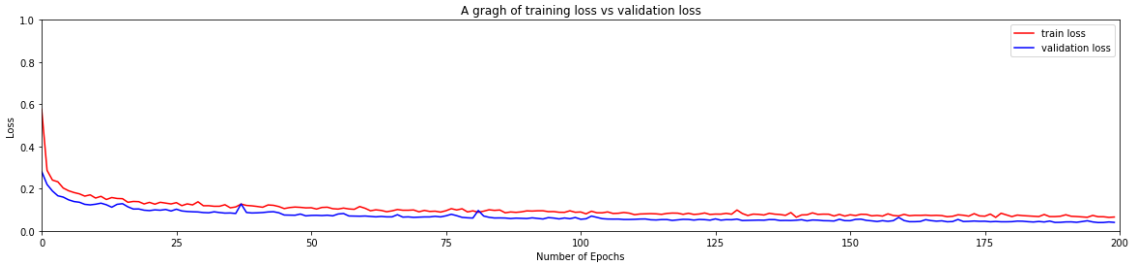


Figure 4.2: Training and validation learning curves from the training of ResNet 50

Figures 4.1 and 4.2 show the learning curves obtained after training the models. In both graphs, the curves show a downward trend, which indicates a decrease of loss obtained in the training and validation images throughout the training epochs. Although they show a positive evolution in the models' performance, the curves present a difference that is important to mention.

At the beginning of the training, even before the 25 training epochs, the loss of validation exceeds the training loss. This may indicate a case of overfitting, as the model has trouble generalizing with new data.

Throughout the training, the evolution of the loss obtained in the validation images follows the loss obtained in the training images, regarding ResNet 50, as seen in figure 4.2. It can also be observed that the training loss is higher than the loss curve obtained with the validation images. Although both curves are converging over time, these results can indicate a case of slight underfitting, since, at the end of the training, the training loss remains higher than validation loss.

However, to evaluate the model's test performance, the remaining metrics considered must also be assessed.

		Number of training epochs							
		25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
ResNet 18	**3 classes**	0.58	0.56	0.57	0.48	0.54	0.53	0.54	0.57
	2 classes	0.66	0.65	0.66	0.59	0.61	0.63	0.60	0.67

Table 4.1: F1-score results obtained with ResNet 18

		Numer of training epochs							
		25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
ResNet 50	**3 classes**	0.62	0.66	0.60	0.60	0.59	0.61	0.56	0.60
	2 classes	0.69	0.74	0.66	0.66	0.66	0.70	0.63	0.67

Table 4.2: F1-score results obtained with ResNet 50

The F1-score test results obtained after training the models can be seen in tables 4.1 and 4.2. For better visualization of the evolution of the models' performance and as a complement to the table, graphs 4.3 and 4.4 is also shown. For both, the F1-score test results presented were obtained every 25 training epochs.

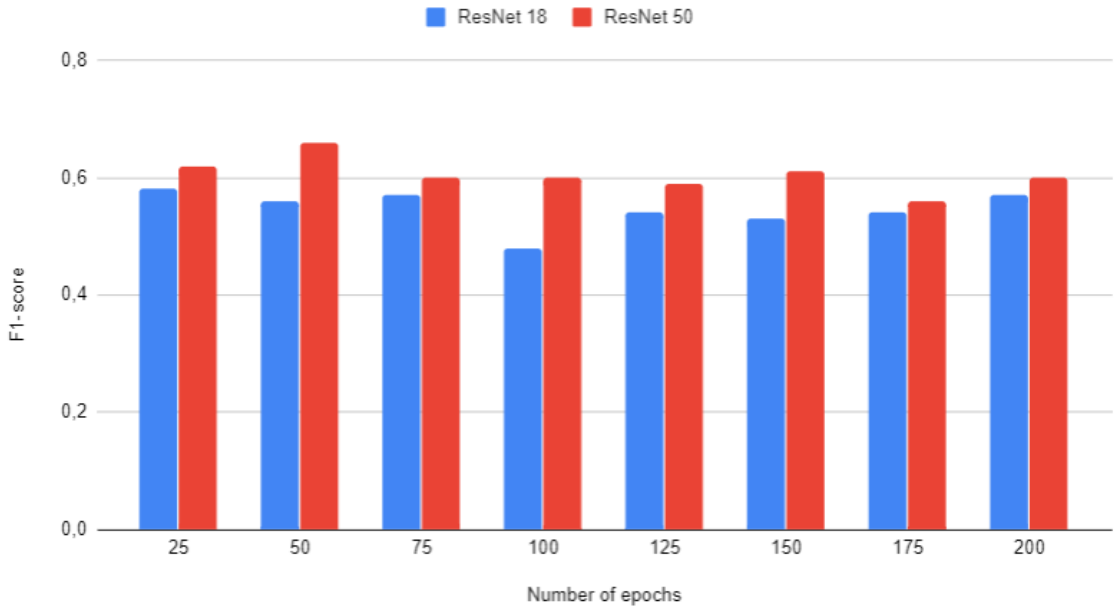


Figure 4.3: F1-score test results every 25 epochs, considering 3 classes

By observing the graphs of the F1-score test results obtained considering three classes, in figure 4.3, it is possible to identify an improvement in the ResNet 50 network performance until epoch 50, with a general stagnation of results from this point forward. In figure 4.4, considering two classes, the F1-score results show a more significant oscillation, which indicates that one of the difficulties of the model when performing classification may be the distinction between fire and smoke. It is also possible to verify, by observing the results obtained considering three classes, that the F1-score test results obtained with the ResNet 18 network appear to slightly worsen after epoch 25 of training.

By analyzing the values presented in table 4.2, it is possible to observe that the results reach a maximum F1-score of 0.66 considering two classes and 0.74 considering three classes, in epoch 50 of training, with the ResNet 50 model. It can also be observed that the values obtained by ResNet 18 reach the maximum F1-score at epoch 25, with some oscillations until the end of the training.

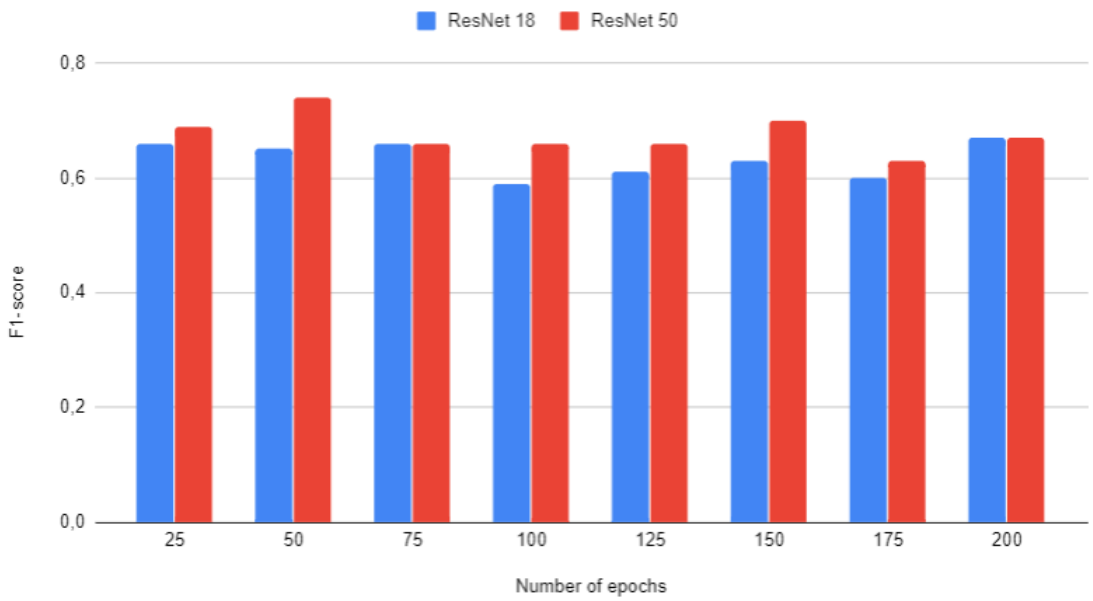


Figure 4.4: F1-score test results every 25 epochs, considering 2 classes

The performance evolution throughout the training phase indicates that the ResNet 18 network takes a smaller number of epochs to converge than ResNet 50. The faster conversion may be due to its simpler architecture, with fewer layers of depth and a smaller number of layers between each "skip connection". Due to this aspect of the ResNets architecture, as expected, after reaching the best F1-score value, as the network training continues, there is no significant worsening of the network performance, only a decreasing trend with the increase in the number of training epochs.

The confusion matrices resulting from the test, obtained by ResNets 18 and 50, considering 3 and 2 classes, are shown in figures 4.5 and 4.6.

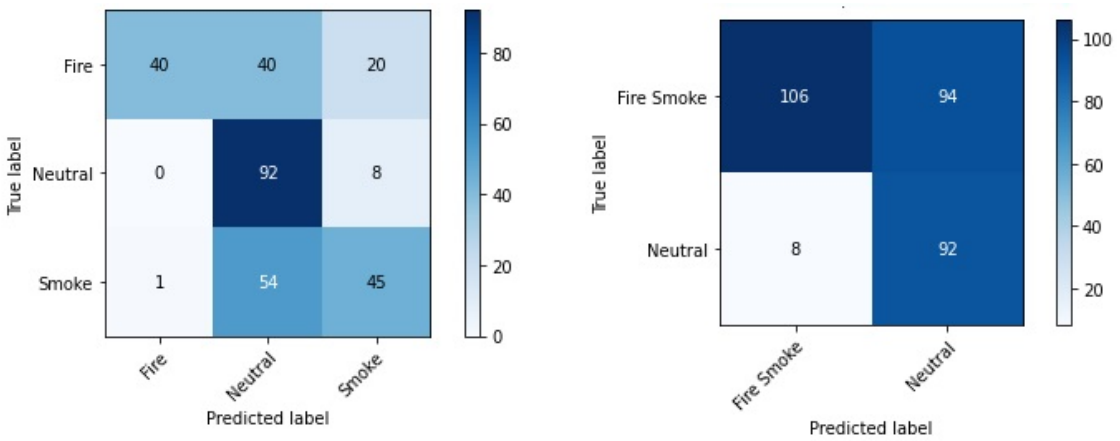


Figure 4.5: Confusion matrices obtained by ResNet 18, considering 3 and 2 classes

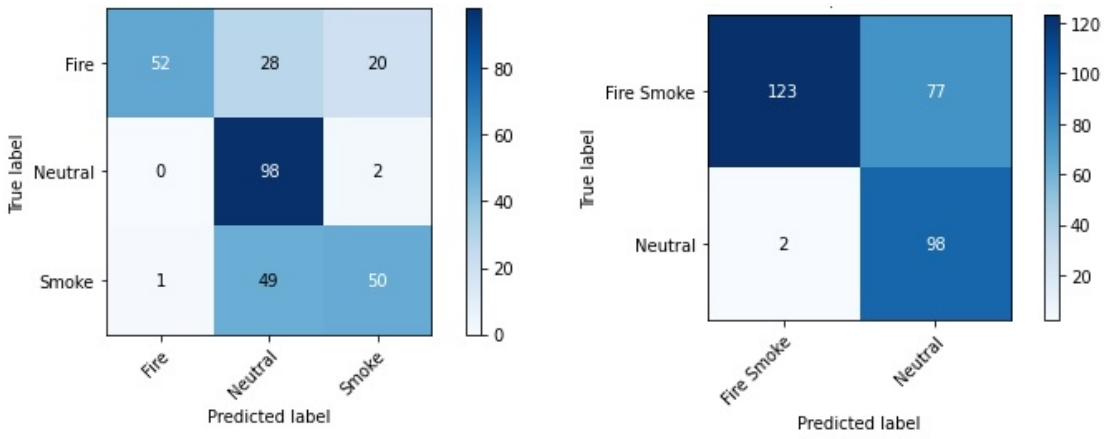


Figure 4.6: Confusion matrices obtained by ResNet 50, considering 3 and 2 classes

In conjunction with the analysis of the learning curves and the F1-score test values obtained by the models, when analyzing the confusion matrices, it is possible to confirm that one of the most significant difficulties is the distinction between the Fire and Smoke classes. Both ResNet 18 and ResNet 50 networks present a large number of images whose class is Fire or Smoke, classified as Neutral, 94, and 77, respectively. These results may mean that the models are prone to false negatives. Nevertheless, comparing the two models, ResNet 18 presents a greater problem in distinguishing between Fire and Neutral classes than ResNet 50.

Both ResNet 18 and ResNet 50 models present the best training F1-score results in the initial training epochs, 25 and 50. In order to improve the results, the learning rate and batch size parameters to be used in the training of the models were then optimized, and dataset augmentation techniques were explored in the following tests. Based on these test results, the ResNet 50 was the network used in those tests.

4.1.1 Training and evaluating the model

This section presents the tests related to the optimization of the model's training parameters. First, the criterion for choosing the model used in the following tests is described. The tests performed to optimize the learning rate, and batch size parameters are presented. Finally, the choice of which values to use to achieve better classification results in the fire and smoke recognition problem is made.

Test for optimization of the parameter: Batch size

The tests were performed with a similar setup as used in the pre-training of these models and consisted of training up to 200 epochs and evaluating the F1-score and confusion matrix test results, only varying the batch size used in each test. The setup for the tests performed is described below:

- Training and validation with *Fire-Smoke-Cropped* dataset:
  - Training images consisted of 70% of the 1000 images in each class (2100 images), of size 224 x 224;
  - Validation images consisted of the remaining 30% of the 1000 training images each class (900 images), also of size 224x224.

- Fixed learning rate value, same as used in the pre-training of the networks: learning rate of 0.001;
- Batch size values tested: 8, 16, 32, 64, 128;
- Use of ResNet 50 model, pre-trained on the *Imagenet* dataset;
- Training up to 200 epochs;
- Tests with images from the *Real-Images-Cropped* dataset: 100 images of each class (300 images) of size 224 x 224.

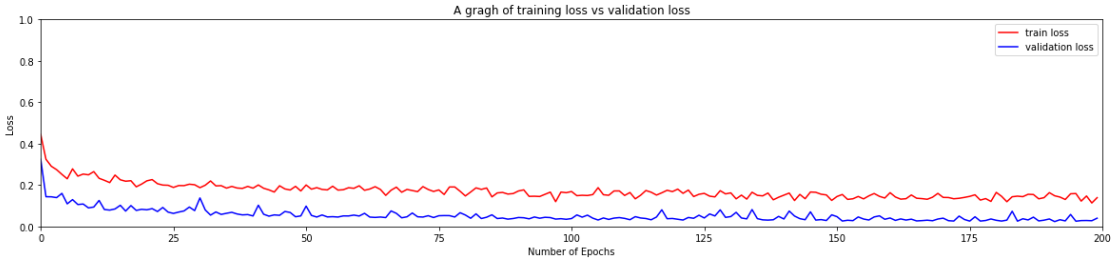


Figure 4.7: Training and validation learning curves with batch size 8

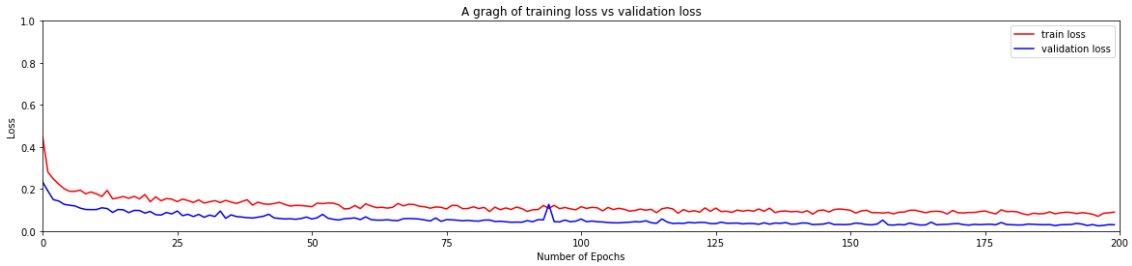


Figure 4.8: Training and validation learning curves with batch size 16

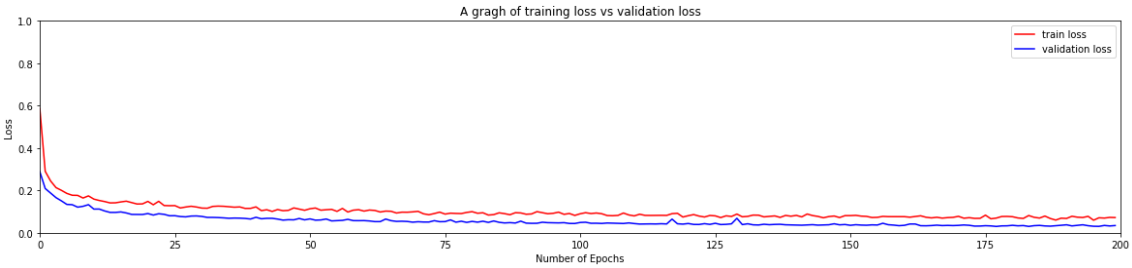


Figure 4.9: Training and validation learning curves with batch size 32

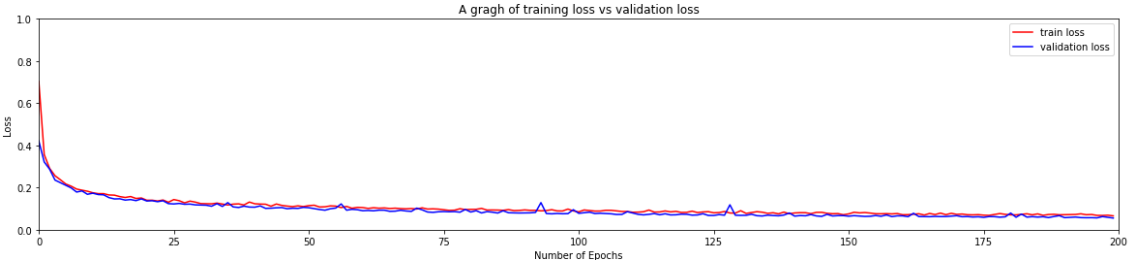


Figure 4.10: Training and validation learning curves with batch size 64

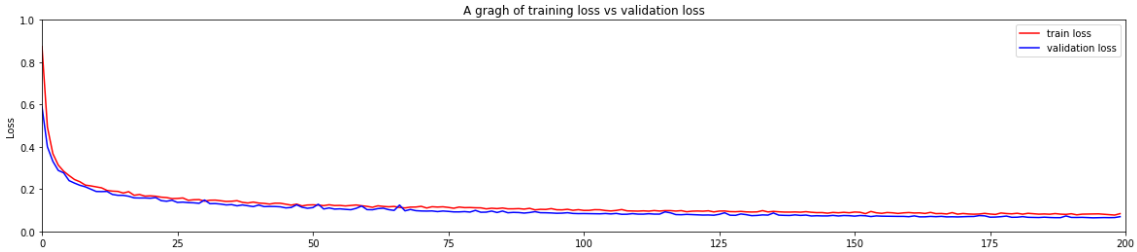


Figure 4.11: Training and validation learning curves with batch size 128

A clear difference can be observed between the curves obtained using batch sizes 8, 16 and 32, and 64 and 128. The first ones present a greater distance between training and validation loss curves. This difference may be an indication of overfitting of the model to the training data.

After comparing all the training loss and validation loss curves obtained using different batch sizes, no further problems occurred during training. Therefore, it is not possible to exclude any of the tested values without an analysis of the models' remaining evaluation metrics.

	Number of epochs			
	50	**100**	**150**	**200**
Batch size 8	0.47	0.53	0.48	0.49
Batch size 16	0.53	0.57	0.57	0.55
Batch size 32	**0.69**	0.67	0.68	0.66
Batch size 64	0.64	0.65	0.64	0.51
Batch size 128	0.67	0.67	0.65	0.65

Table 4.3: F1-scores obtained with different batch sizes every 50 epochs, for 3 classes

	Number of epochs			
	50	**100**	**150**	**200**
Batch size 8	0.50	0.57	0.52	0.53
Batch size 16	0.62	0.63	0.63	0.60
Batch size 32	**0.74**	0.72	0.73	0.71
Batch size 64	0.72	0.73	0.70	0.56
Batch size 128	0.74	0.74	0.72	0.71

Table 4.4: F1-scores obtained with different batch sizes every 50 epochs, for 2 classes

The confusion matrices corresponding to the best F1-score test results obtained, considering two and three classes, are shown in figure 4.12.

The best F1-score test results were 0.69 considering three classes with a correspondent 0.74 considering two classes, as observed in tables 4.3 and 4.4. These results were obtained using batch size 32, at 50 epochs of training.

When observing the best results' confusion matrix calculated with three classes obtained, it can be seen that there is a visible confusion between the Smoke and Neutral classes. About half of the **Smoke** images are classified as belonging to class **Neutral**.

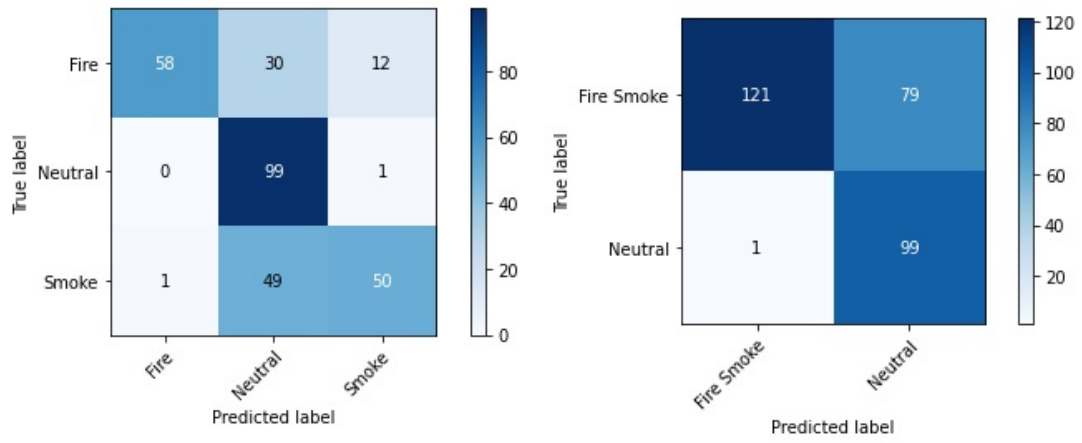


Figure 4.12: Best results confusion matrices considering 2 and 3 classes

By observing the confusion matrix calculated considering two classes, on the right in figure 4.12, it is possible to identify some confusion between the **Fire Smoke**, and **Neutral** classes, with a total of 80 incorrectly classified images, of which 79 would be false negatives in a real context of use. These false negative values are somewhat worrying since they could result in reported fires that would not get any response.

After analyzing the results, the batch size value chosen for the tests that followed was 32. However, due to the model's difficulties in identifying fire and smoke observed, it was necessary to perform additional tests to improve the models' performance.

Similarly to the previous test, without any form of data augmentation the performance of the models has a fast conversion and, in general, it stagnates between 50 and 100 epochs of training.

Test for optimization of the parameter: Learning rate

The training setup was similar to the previous tests, only varying the learning rate parameter used in each test:

- Training and validation with *Fire-Smoke-Cropped* dataset:
  - Training images consisted of 70% of the 1000 images in each class (2100 images), of size 224 x 224;
  - Validation images consisted of the remaining 30% of the 1000 training images each class (900 images), also of size 224x224.
- Fixed batch size and learning rate values, as used in the pre-training of the networks: batch size 32;
- Learning rate values tested: 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001;
- Use of ResNet 50 model, pre-trained on the *Imagenet* dataset;
- Training up to 200 epochs;
- Tests with images from the *Real-Images-Cropped* dataset: 100 images of each class (300 images) of size 224 x 224.

Figures 4.14, 4.15, 4.16, 4.17 and 4.18 show graphs of the evolution of training and validation loss over the models' training epochs, considering different learning rates. The loss values obtained with the learning rate of 0.0000001 were too high to present a similar graph, and, as such, this value can be excluded as a possibility to be used in training the models to improve test performance.

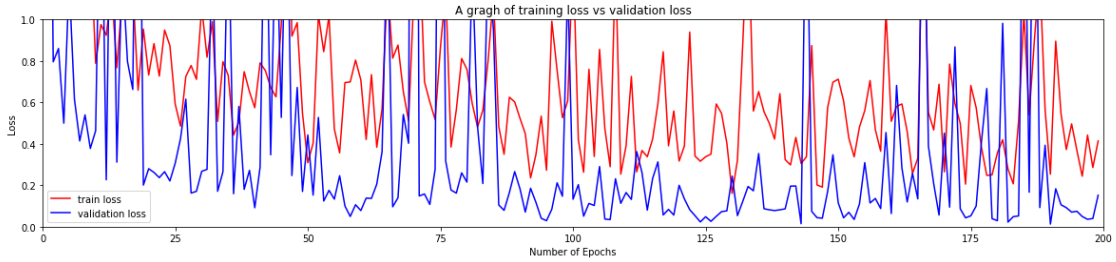


Figure 4.13: Training and validation learning curves with learning rate 0.1

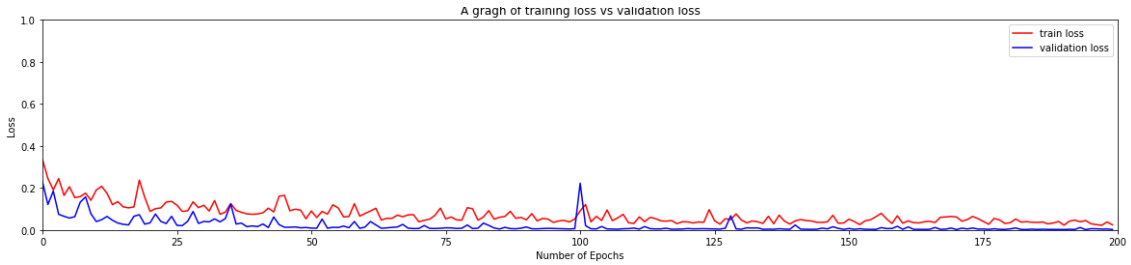


Figure 4.14: Training and validation learning curves with learning rate 0.01

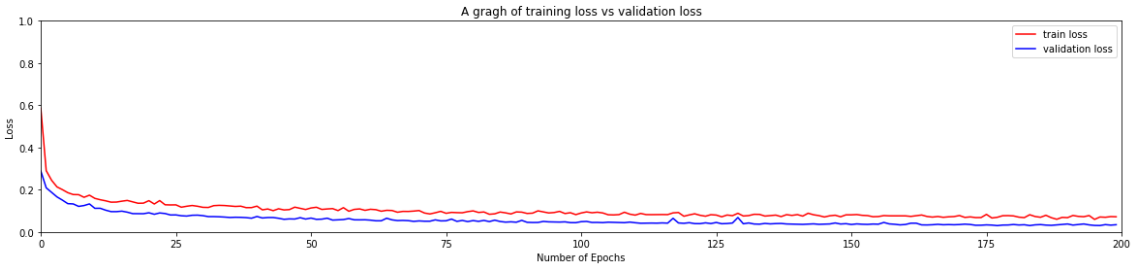


Figure 4.15: Training and validation learning curves with learning rate 0.001

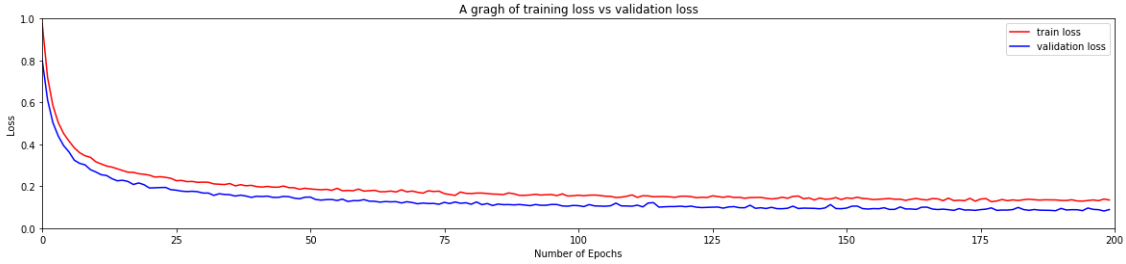


Figure 4.16: Training and validation learning curves with learning rate 0.0001

The analysis of these graphs shows that when using 0.1 (figure 4.14) or 0.01 (figure 4.15) learning rates, the loss values obtained show a more significant oscillation than with the other tested values. The value of learning rate 0.1 can be excluded, since, by being too high, it does not allow convergence to a lower loss.

The graph resulting from the training with the learning rate 0.01 shows a convergence of loss results that is too fast and, as such, it may also not be a good choice for the learning

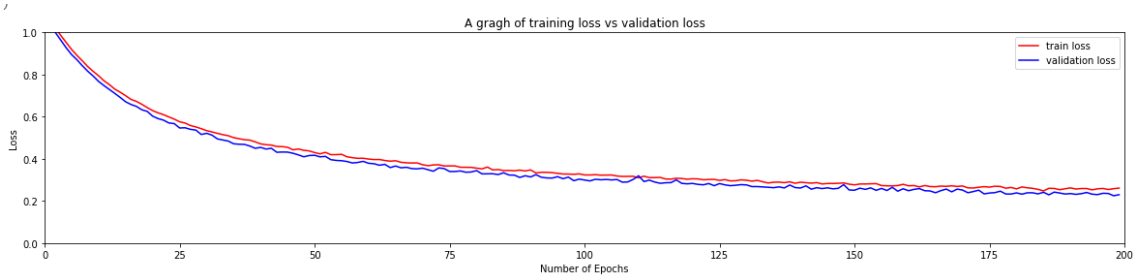


Figure 4.17: Training and validation learning curves with learning rate 0.00001

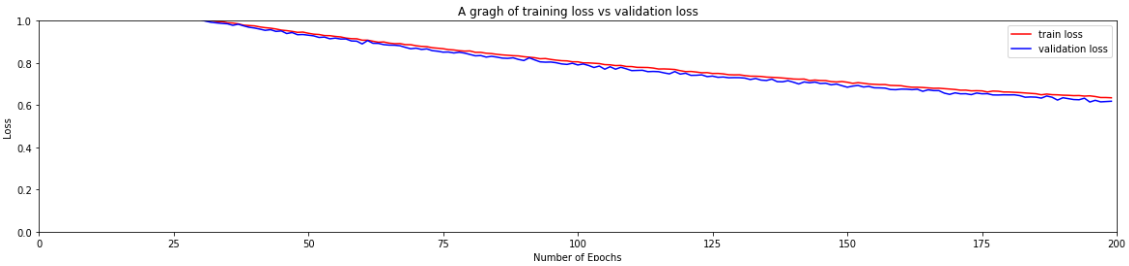


Figure 4.18: Training and validation learning curves with learning rate 0.000001

rate value. On the other hand, the graph resulting from training with a learning rate of 0.000001 (figure 4.18) shows the opposite situation. Although both curves have a downward trend, the evolution of training and validation loss may be too slow for the convergence of the models to be possible.

For the choice of the learning rate, complementing the analysis of training and validation loss curves, the F1-score test values obtained by the models trained with the different learning rates were also taken into account. These values are presented in tables 4.5 and 4.6, calculated considering three and two classification classes.

	Epochs of training			
Learning Rate	**50**	**100**	**150**	**200**
0.0000001	0.27	0.32	0.35	0.35
0.000001	0.47	0.53	0.56	0.53
0.00001	0.53	0.55	0.57	0.58
0.0001	0.69	0.72	0.61	0.64
0.001	0.69	0.67	0.68	0.66
0.01	0.64	0.48	0.54	0.53
0.1	0.63	0.49	0.56	0.51

Table 4.5: F1-score test results varying learning rate considering 3 classes

The tables with the F1-score test results obtained after the models' training show that the best result was 0.72 considering two classes and 0.82 considering three classes. This result was obtained using a learning rate of 0.0001 after 100 epochs of training.

The test confusion matrices obtained considering two and three classes corresponding to the best F1-score results are shown in figure 4.19.

	Epochs of training			
Learning rate	**50**	**100**	**150**	**200**
0.0000001	0.50	0.51	0.52	0.53
0.000001	0.56	0.63	0.64	0.62
0.00001	0.57	0.69	0.69	0.71
0.0001	0.79	**0.82**	0.69	0.71
0.001	0.74	0.72	0.73	0.71
0.01	0.70	0.53	0.62	0.60
0.1	0.67	0.63	0.69	0.56

Table 4.6: F1-score test results varying learning rate considering 2 classes

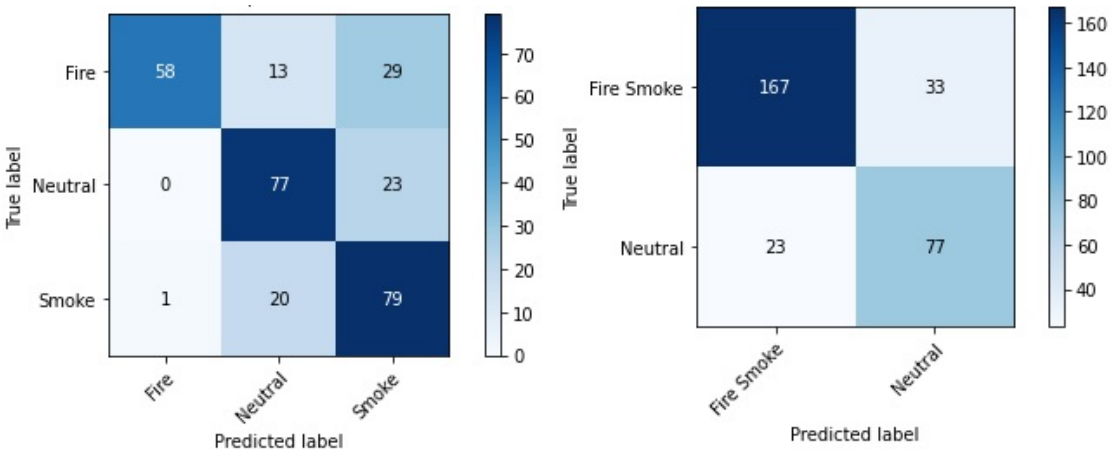


Figure 4.19: Best result confusion matrices considering 2 and 3 classes

Compared to the models previously analyzed, it presents only a total of 56 images classified incorrectly. Only 33 correspond to images where signs of fire or smoke are present and were classified as **Neutral** and, as such, would result in false negatives in a real context of use.

These results indicate that the use of learning rate 0.0001 improves the classification performance of the models, and, as such, it was the value chosen to carry out the tests that followed.

4.1.2 Data pre-processing

The models were trained using the dataset *Fire-Smoke-Cropped* and tests were performed using the dataset *Real-Images-Cropped*. In this section the tests regarding data pre-processing for trainig the models is addressed.

The tests performed regarding the pre-processing of the images took into account dataset augmentation techniques, ensuring that the images belonged to the same class after the transformations were applied. As such, only the application of rotations and horizontal and vertical flip of the images were considered. These tests were performed after the optimization of the leaning rate and batch size parameters for training ResNet networks.

Test with all ResNets, using data augmentation

The first tests performed using data augmentation techniques aimed to compare all ResNet

models' performance, using 90º rotation and random horizontal amd vertical flip of images as pre-processing and, therefore, had the following setup:

- Training and validation with *Fire-Smoke-Cropped* dataset:
  - Training images consisted of 70% of the 1000 images in each class (2100 images), of size 224 x 224;
  - Validation images consisted of the remaining 30% of the 1000 training images each class (900 images), also of size 224x224.
- Use of dataset augmentation techniques that allowed the images to remain class-persistent: image rotation of 90º and random horizontal and vertical flip of the images;
- Fixed batch size and learning rate values, previously optimized: batch size of 32, and learning rate of 0.0001;
- Use of all ResNet model variations considered: ResNets 18, 34, 50, 101 and 152, pre-trained on the *Imagenet* dataset;
- Training up to 200 epochs;
- Tests with images from the *Real-Images-Cropped* dataset: 100 images of each class (300 images) of size 224 x 224.

Tables A and B show the F1-score values of the test obtained, considering two and three classes.

Model	**Number of epochs**							
	25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
ResNet 18	0.58	0.63	0.63	0.64	0.60	0.59	0.61	0.59
ResNet 34	0.51	0.54	0.51	0.49	0.49	0.55	0.47	0.52
ResNet 50	0.54	0.62	0.66	0.65	0.68	0.68	0.68	**0.71**
ResNet 101	0.61	0.65	0.65	0.65	0.71	0.61	0.67	0.67
ResNet 152	0.61	0.64	0.65	0.64	0.60	0.70	0.60	0.66

Table 4.7: F1-score test results obtained with other models considering 3 classes

Model	**Number of epochs**							
	25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
ResNet 18	0.64	0.73	0.71	0.73	0.69	0.69	0.70	0.68
ResNet 34	0.61	0.62	0.59	0.56	0.56	0.62	0.53	0.60
ResNet 50	0.70	0.74	0.79	0.79	0.79	0.79	0.79	**0.83**
ResNet 101	0.67	0.71	0.71	0.71	0.75	0.67	0.62	0.72
ResNet 152	0.66	0.70	0.69	0.71	0.65	0.75	0.66	0.73

Table 4.8: F1-score test results obtained with all ResNet models considering 2 classes

The results obtained with ResNet 50 stand out. Although there is a stagnation of results between epochs 125 and 175 of training, in the end, there is an improvement between epochs 175 and 200.

The best F1-score test result was 0.71 considering three classes, corresponding to 0.83 considering two classes. These results were obtained with ResNet 50, at 200 training epochs. The training loss and validation curves resulting from the training of the ResNet 50 are shown in figure 4.20. Since best results were obtained with ResNet 50 at 200 training epochs, the corresponding test confusion matrices are also shown, in figure 4.21.

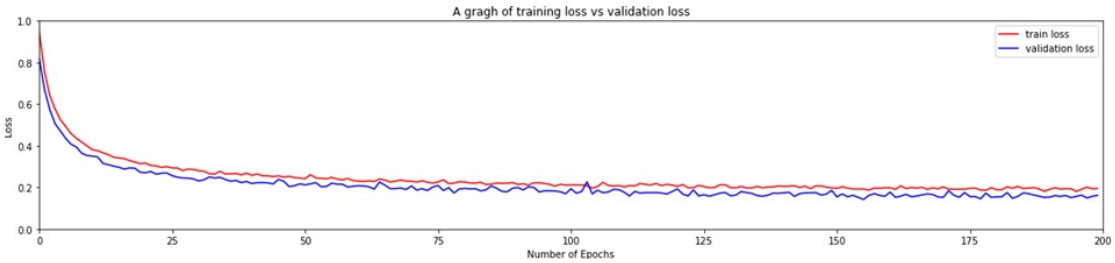


Figure 4.20: ResNet 50 training and validation learning curves

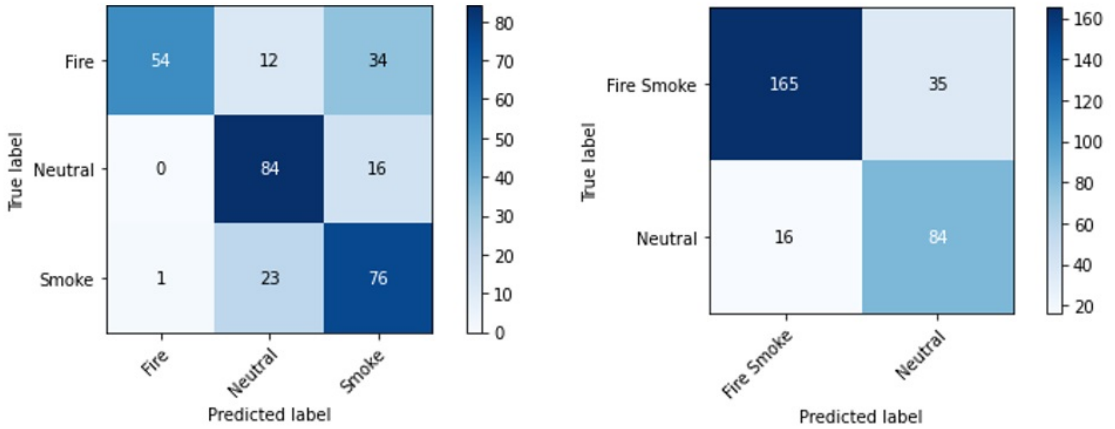


Figure 4.21: Best result confusion matrices considering 2 and 3 classes

The results presented in the confusion matrices indicate that there is still considerable confusion between the classes **Fire Smoke** and **Neutral**, resulting in a worrying number of false positives. However, in general, comparing the results with those of the previous tests, the F1-score values obtained by the different ResNet models show a more significant oscillation throughout the training epochs. This difference can be justified with the use of data augmentation techniques in training, which prevent the early stagnation of results.

In addition, the graph of the learning curves with the evolution of training loss and validation throughout the training of the ResNet 50, indicates that there is no overfitting of the models. Once the best result was obtained at 200 training epochs, a second test was carried out, considering a larger number of training epochs, in order to try to improve the classification performance.

Test with ResNet 50, using data augmentation

According to the results obtained in the previous test, a test was performed, using the same data augmentation techniques, with the ResNet 50 model, which obtained better performance compared to the other models. This test consisted of using ten different trained models using the same setup, averaging the F1-score results obtained. When evaluating the average results, taking into account the randomness of the applied transformations, it is possible to draw more objective conclusions about the generalization capacity of the models.

Each run had the following setup:

- Training and validation with *Fire-Smoke-Cropped* dataset:
  - Training images consisted of 70% of the 1000 images in each class (2100 images), of size 224 x 224;
  - Validation images consisted of the remaining 30% of the 1000 training images each class (900 images), also of size 224x224.
- Use of the same dataset augmentation techniques as in the previous test: image rotation of 90º and random horizontal and vertical flip of the images;
- Fixed batch size and learning rate values, previously optimized with the tests presented in the following section: batch size of 32 and learning rate of 0.0001;
- Use of the ResNet model that obtained best results in the previous test: ResNet 50, pre-trained on the *Imagenet* dataset;
- Training up to 1000 epochs;
- Tests with images from the *Real-Images-Cropped* dataset: 100 images of each class (300 images) of size 224 x 224.

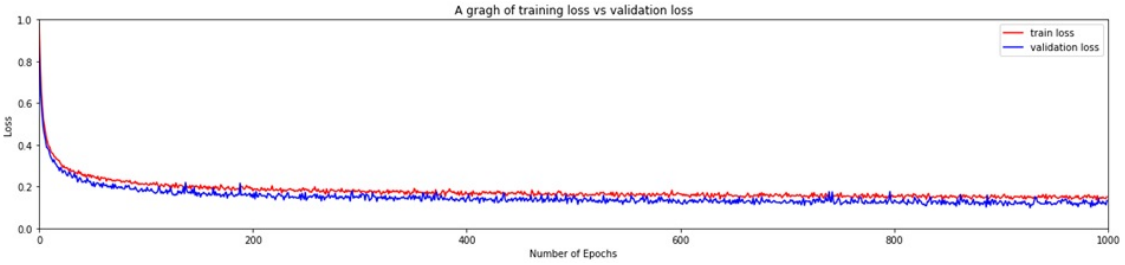


Figure 4.22: Best result raining and validation learning curves up to 1000 epochs

In figure 4.23 is a graph with the evolution of F1-score test results obtained throughout the models' training epochs and tables 4.9 and 4.10 present the average of F1-score test results obtained using as data augmentation 90º rotation and horizontal and vertical flip of the cropped images. Comparing these results to the previous tests, the F1-scores obtained considering two classes are much higher than the original ones, considering three classes. This result shows that the models still have some difficulties in distinguishing between the **Fire** and **Smoke** classes when trained with data augmentation techniques.

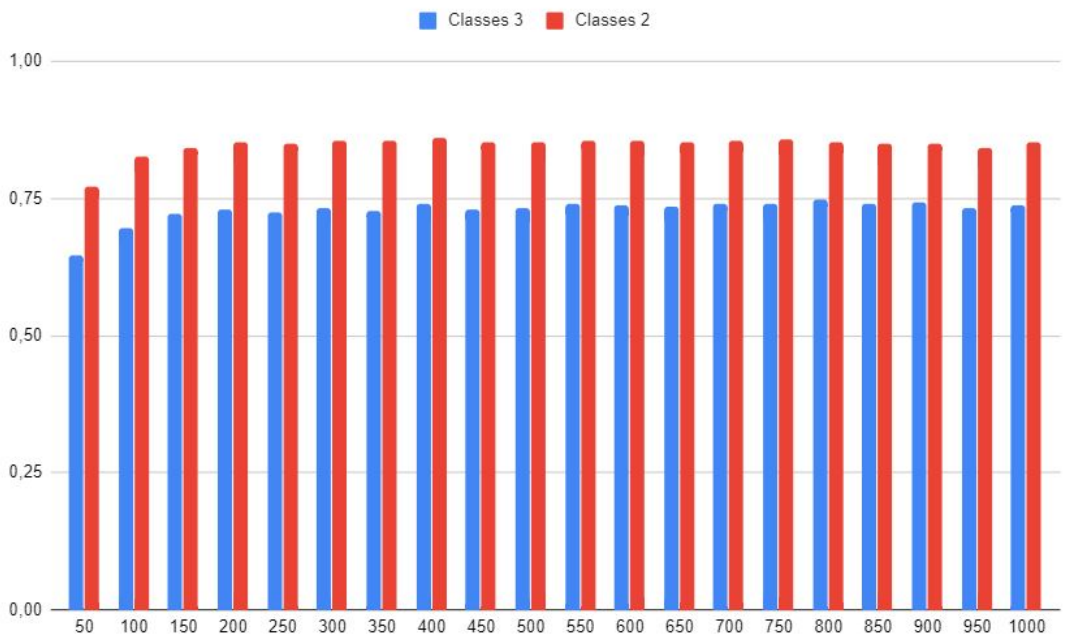


Figure 4.23: F1-score average test results obtained every 50 epochs of training

	Number of epochs									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
3 classes	0.68	0.71	0.72	0.74	0.75	0.73	0.73	0.74	0.71	0.71
2 classes	0.78	0.83	0.84	0.84	0.85	0.86	0.85	0.85	0.85	0.83

Table 4.9: ResNet 50 F1-score average test results, every 50 epochs of training (up to 500)

	Number of epochs									
	550	**600**	**650**	**700**	**750**	**800**	**850**	**900**	**950**	**1000**
3 classes	0.74	0.74	0.74	0.75	0.74	0.74	0.74	0.73	0.74	0.74
2 classes	0.85	0.85	0.85	0.86	0.85	0.86	0.86	0.85	0.85	0.84

Table 4.10: F1-score average test results obtained with ResNet 50 every 50 training epochs

In appendix 5 the tables with F1-score test results referring to each of the 10 runs are presented, considering three and two classes. The best F1-score results were obtained after 550 epochs of training. This model reaches an F1-score of 0.76 considering two classes and a corresponding 0.88 considering two classes.

Comparing these results to the previous tests, the F1-scores obtained considering two classes are much higher than the original ones, obtained considering three classes. This shows that the model still has some difficulties in distinguishing between the **Fire** and **Smoke** classes. Nevertheless, the results obtained with both two and three classes considered show an improvement in test performance of the models with the use of dataset augmentation techniques in training.

Referring to this model, table 4.11 presents the F1-score test results per class and confusion matrices considering three and two classes are presented in figure 4.24.

	F1 score
Fire	0.81
Smoke	0.76
Neutral	0.76

Table 4.11: Best F1-score results (obtained at 550 training epochs)

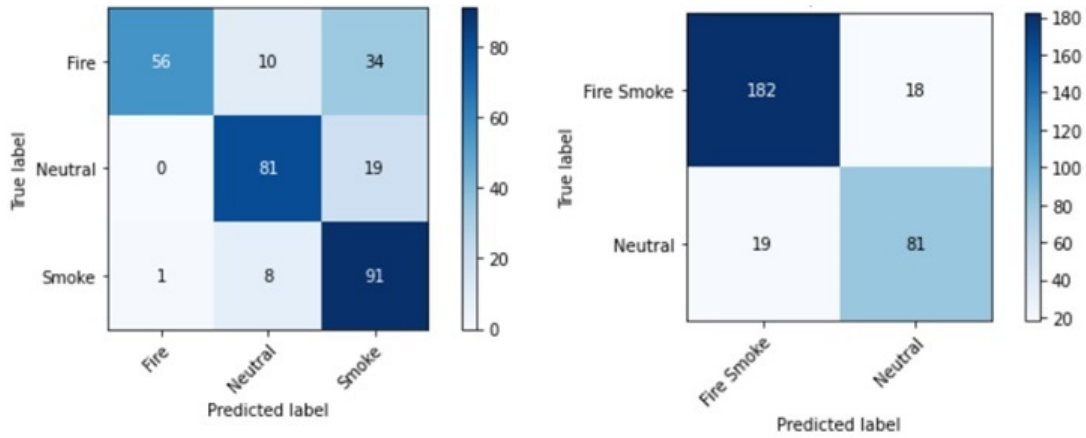


Figure 4.24: Confusion matrix obtained at 550 training epochs, considering 2 and 3 classes

The confusion matrices allow us to verify that the model becomes less prone to false negatives when using data augmentation techniques in training. There is less confusion between images belonging to the **Fire** or **Smoke** class and the **Neutral** class (only 18 images were confused). However, as observed in the analysis of the average F1-score results in graph the 4.9, there is still significant confusion between the classes of smoke and fire, resulting in another 35 images classified incorrectly.

4.1.3 Discussion of Classification Results

The ResNet networks were chosen for the development of the image classification approach, using pre-trained models with the *Imagenet* dataset. The tests carried out as part of the development of this approach are summarized in table 4.12.

First, tests were performed to choose the model for hyper-parameter optimization, as suggested in [66]. Then, with the hyper-parameters considered already optimized, tests were performed to improve the models' classification performance using data augmentation techniques.

Considering the optimized hyper-parameters, the performances of all ResNets (18, 34, 50, 101, and 152), using data augmentation techniques, were compared. With the model that obtained the best results, another set of tests was performed, with a higher number of training epochs, to improve the models' classification performance using data augmentation techniques.

Test name	Test performed	Test objective	Results evaluation metrics
Initial tests	Performance comparison between the models considered in this approach (ResNet 18 and ResNet 50) for the best to be used in the following tests	Choosing the model to be used in the following tests	Training and validation learning curves, F1-score test results and confusion matrices obtained for 3 and 2 classes
Test for batch size optimization	Performance comparison between the ResNet 50 models trained with different hyper-parameter values considered	Optimize training hyper-parameters to obtain better image classification	Training and validation learning curves, F1-score test results and confusion matrices obtained for 3 and 2 classes
Test for learning rate optimization			
Test with all ResNets, using data augmentation	Performance comparison between all considered ResNet models, trained with hyper-parameter values optimized in previous tests	Use data augmentation to obtain better image classification	F1-score test results obtained with all ResNets, training and validation learning curves and confusion matrices obtained considering 3 and 2 classes
Test with ResNet 50, using data augmentation	Training model with the best result in the previous test (ResNet 50) up to 1000 epochs of training, presenting an average of 10 runs	Continue training best model obtained in the previous test to obtain better image classification	Training and validation learning curves, F1-score average test results, confusion matrices obtained considering 3 and 2 classes

Table 4.12: Tests performed for the image classification approach

The models' training parameters batch size, and learning rate, were optimized, adapting them to the fire and smoke recognition problem in static images.

The best results obtained with ResNet 50, using dataset augmentation techniques (ensuring that the image belonged to the same class), reaching the values of F1-score test of 0.76 considering three classification classes, **Fire**, **Smoke**, and **Neutral**, and 0.88 considering two classes, **Fire Smoke** and **Neutral**. However, even using transfer learning and data augmentation techniques in the models' training, there are still some problems encountered with the classification approach.

The results indicate that the models have problems in distinguishing between fire and smoke in the images. These difficulties in classification may be due to the fact that there are fire and smoke simultaneously in most of the examples belonging to class **Fire** (similar to real photographs of forest fires that will be submitted for classification).

As a result, the image object detection approach was studied as an alternative, whose tests are presented in the following section.

4.2 Results obtained with YOLO networks

This section presents the different tests carried out within the scope of the development of the image object detection approach. The tests that aimed to improve the detection performance of the models are presented, varying the pre-processing of the images and the optimization of some training parameters of the You Only Look Once (YOLO) models.

Initial tests to evaluate the models' performance

The initial tests aimed to understand and analyze the performance of the different models considered. The tests were performed to compare the YOLOv3 and YOLOv4 models' performance.

For these tests, the default values recommended for training the models, as specified in works [57] and [22], were considered. Both training an testing datasets were manually annotated, and, as such, these tests also allowed to confirm the possibility of using them for training and testing YOLO models.

The initial tests setup can, therefore, be described as follows:

- Training with *Fire-Smoke-YOLO* dataset;
- Pre-trained networks with *Imagenet* dataset;
- Number of classes = 2 (since the objects considered for detection are **Fire** and **Smoke**);
- Channels = 3 (RGB);
- Training up to 6000 epochs;
- Default input size 416 x 416;
- Training of YOLO models considered: YOLOv3 and YOLOv4;
- Use of default anchors:
  - For YOLOv3:
    anchors = (10,13), (16,30), (33,23), (30,61), (62,45), (59,119), (116,90), (156,198), (373,326);
  - For YOLOv4:
    anchors = (12, 16), (19, 36), (40, 28), (36, 75), (76, 55), (72, 146), (142, 110), (192, 243), (459 , 401).
- Testing with the *Real-Images-YOLO* dataset;

Both the training and testing datasets used in the initial tests are described in table 4.13.

	Total Fire examples	**Total Smoke examples**
Train dataset	2598	2348
Test dataset	460	421

Table 4.13: Number of examples present in training and testing datasets

The mAP test results obtained with the trained YOLOv and YOLOv4 models are shown in figure 4.25. Figures 4.26 and 4.27 show the AP(Fire) and AP(Smoke) results obtained by each of the models.

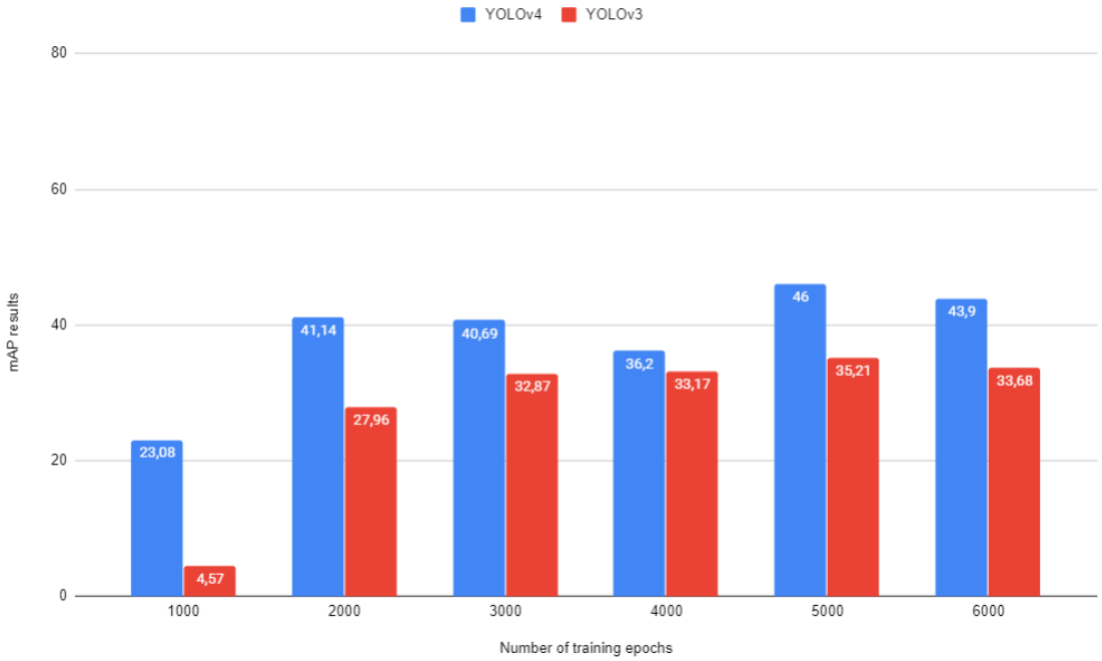


Figure 4.25: mAP test results

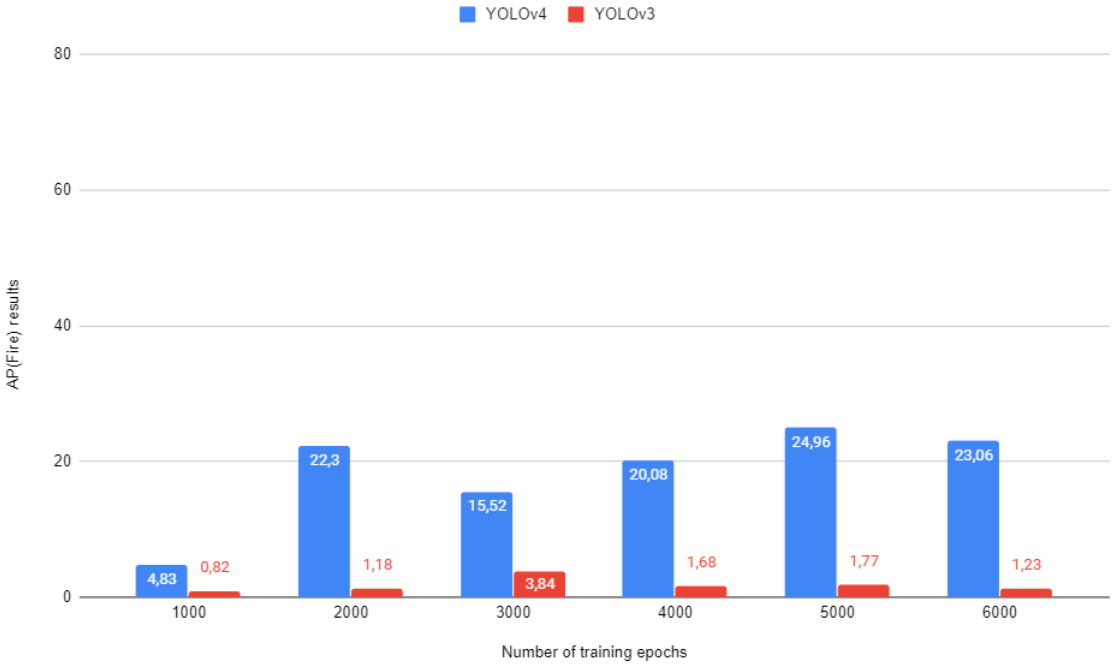


Figure 4.26: AP test results for class Fire

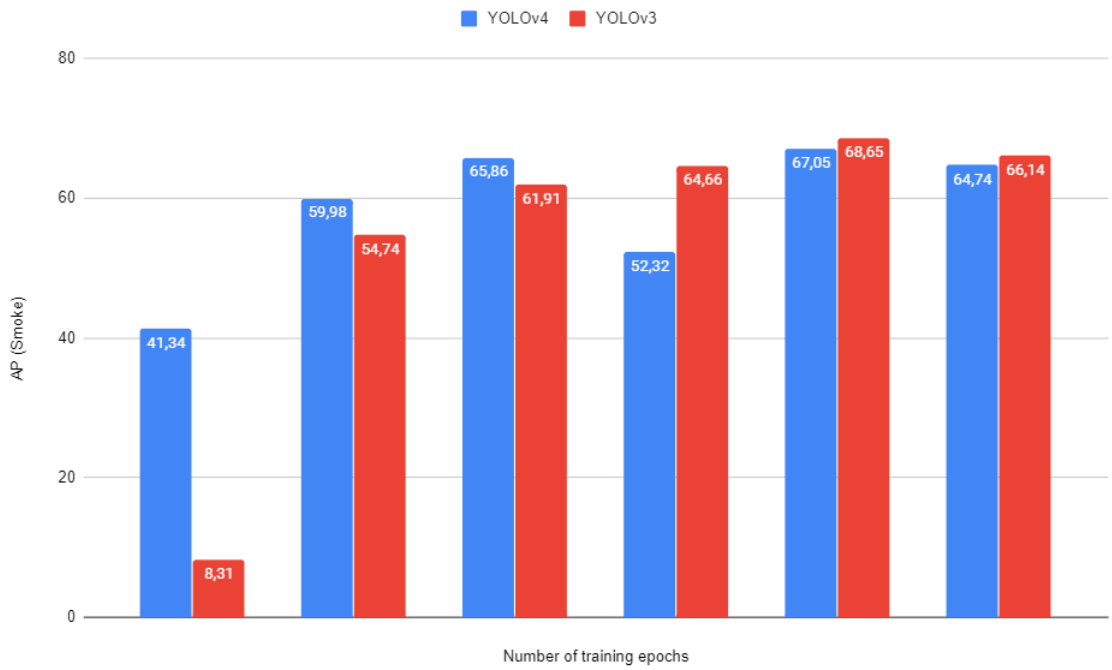


Figure 4.27: AP test results for class Smoke

As shown in figure 4.25, the YOLOv4 model obtains better mAP test results than YOLOv3. YOLOv4 can achieve a test mAP test result of 46%, after 5000 training epochs, while YOLOv3's best result is only of 35.21%, also after 5000 epochs of training.

The AP results for each class obtained by the models, in figures 4.26 and 4.27, show that both models have a better performance detecting objects of class Smoke. The best AP result regarding class Smoke was 68.65%, obtained by YOLOv3, better than the best YOLOv4 AP(Smoke) test result, 67.05%, as observed in figure 4.27.

Although the YOLOv3 model obtains better results in the detection of smoke, it presents serious difficulties in the detection of Fire, obtaining a maximum AP(Fire) value of 3.84% at 3000 training epochs. The YOLOv4 model achieves a better result, reaching a higher value of 24.96%, as shown in figure 4.26.

Between the two tested models, YOLOv4 obtained better results, being, therefore, the chosen one to be used in the following tests to adjust the training anchors.

Test performed with anchor adjustment

The definition of the anchors used in training is an important aspect since it directly impacts the detections made by the models. By adjusting the sizes of the anchors used in training, it will be possible to obtain a better location of the fire and smoke in the images, resulting in an improvement in the results of the mean Average Precision (mAP) test.

This test aimed to compare the trained models' performance with the anchor boxes' definition by assigning them default values, with the performance achieved using k-means clustering to define anchor values adapted to the test dataset. The test setup is described below:

- Training with *Fire-Smoke-YOLO* dataset;
- Use of pre-trained networks with *Imagenet* dataset;

- Number of classes = 2;
- Channels = 3 (RGB);
- Training up to 6000 epochs;
- Default input size 416 x 416;
- Training of YOLOv4 model with adjusted anchors, comparing the results with the ones obtained in the previous test;
- Use of anchor boxes adjusted to the training dataset using k-means
  - For YOLOv4:
    anchors = (35, 37), (60, 84), (144, 89), (90,160), (200,167), (134,256), (334,188), (233,290), (375,342)
- Testing with the *Real-Images-YOLO* dataset;

As in the previous test, both the training and testing datasets are described in table 4.13, presented above.

The mAP test results obtained with the trained YOLOv4 models, using the default and adjusted anchors, are shown in figure 4.28. Figures 4.29 and 4.30 show the AP (Fire) and AP (Smoke) results obtained by each of the models.

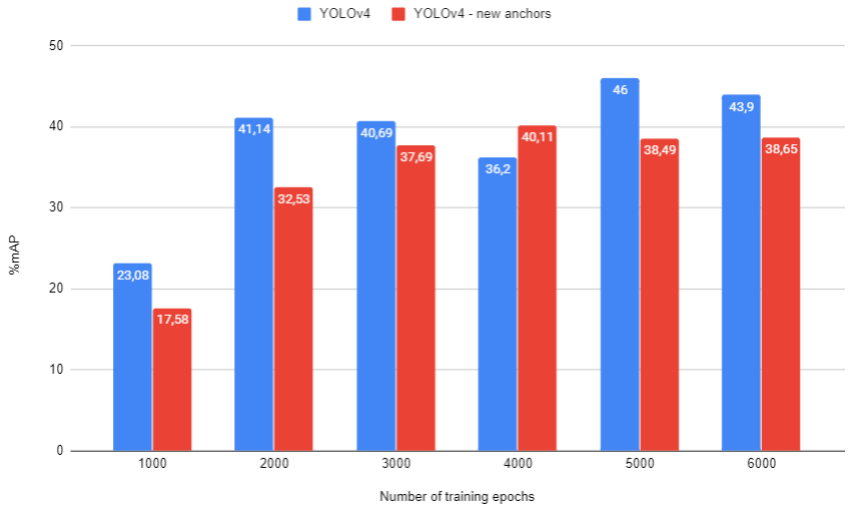


Figure 4.28: mAP test results

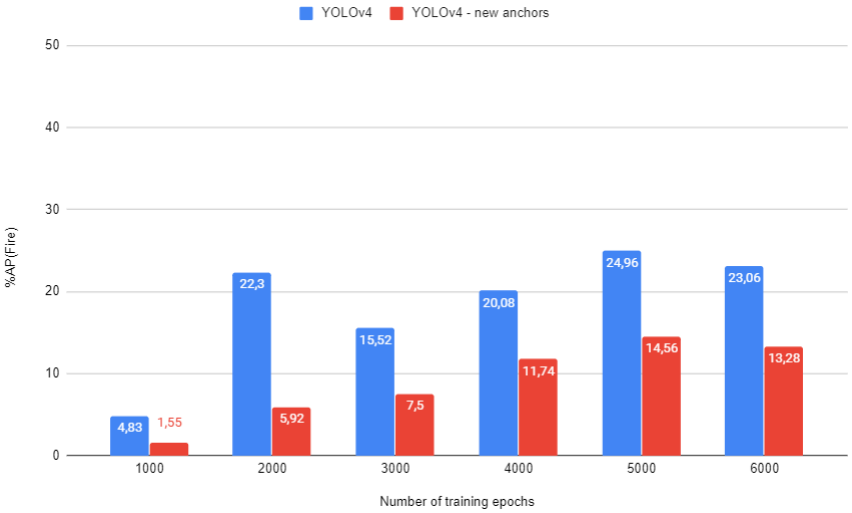


Figure 4.29: AP test results for class Fire

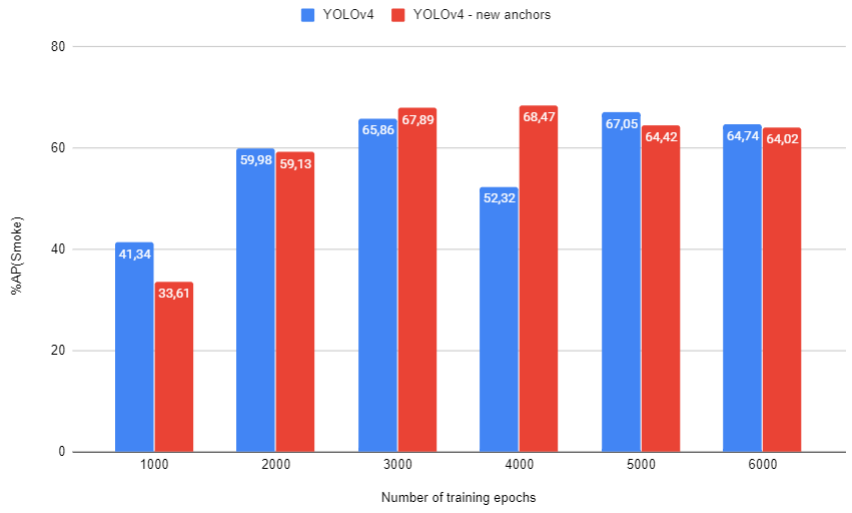


Figure 4.30: AP test results for class Smoke

With the analysis of the mAP results, in figure 4.28, it is possible to conclude that the YOLOv4 model trained with the default anchors obtains better mAP test results in the images' fire and smoke detections. While the model trained with the default anchors can achieve a maximum mAP result of 46%, after 5000 training epochs, the adjusted anchors model's best result is only 40.11%, at 3000 epochs.

The AP results for each class obtained by the models show a significant discrepancy between the models' ability to detect fire in the images. Although both have managed to obtain good results in detecting smoke in the images, the model trained with adjusted anchors is able to surpass the best AP(Smoke) result, reaching 68.47% (in figure 4.30). The worst mAP results obtained with the anchor adjustment are due to the model's difficulty in detecting fire in the images, as can be seen in the graph present in figure 4.26.

In general, the model that uses the default anchor values is able to obtain superior AP(Fire) results throughout the training epochs, which suggests that the anchor adjustment may have been poorly performed. One factor that may have negatively influenced the anchor adjustment is the difference between the type of images used for training and the images used for testing.

In order to approximate the characteristics of the training images to the test images, additional tests were performed, with different combinations between the two datasets. The hypothesis tested will be that by adding images that contain fire and smoke in dimensions similar to those found in forest fire situations, it will be possible to make a better adjustment of the anchors and, consequently, obtain better results in the detections.

4.2.1 Data pre-processing

The tests performed regarding the pre-processing of the images in the object detection approach consisted of the variation of the data used for training. After optimizing the parameters used for the configuration of the YOLO models, taking into account the differences in the context of both datasets' images, different combinations of the annotated images were tested for training the model, setting a new test dataset. These tests were carried out in order to improve the performance of the YOLOv4 models, using different sets of images to adjust the anchors. The hypothesis tested is that, by also using images with specific characteristics observable in situations of forest fires for training, such as those of the test dataset, it is possible to better approximate the dimensions of the anchor boxes to those of the objects to be detected.

The new test dataset consists of half of the *Real-Images-YOLO* dataset, making a total of 215 annotated images. The dataset variations used in the different tests are described in table 4.14. To better understand the differences between the new datasets considered, the graph 4.31 is presented as a complement to the table.

Tests for adjusting anchors, considering different training datasets

Dataset	Total examples of Fire	Total examples of Smoke
New test dataset	238	213
Original training dataset	2598	2348
Original training dataset with half of the original test dataset	2820	2556
Test images used for training	222	208

Table 4.14: Dataset variations used for YOLO tests

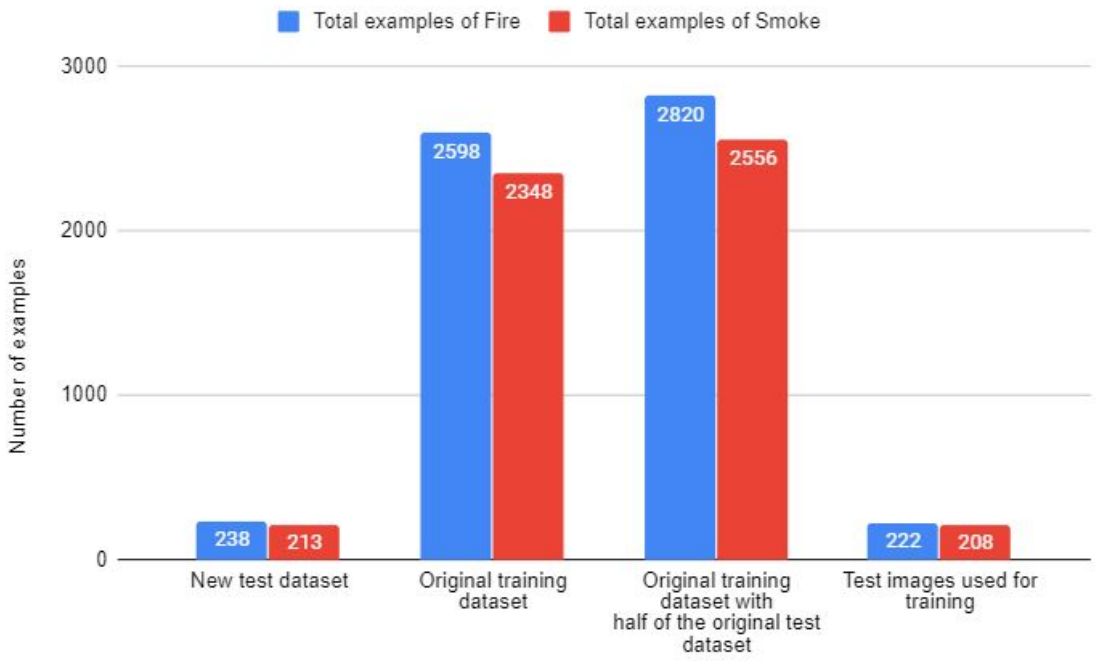


Figure 4.31: Graph with dataset variations used for anchor adjustment tests

These tests were carried out in order improve the detection performance, by adjusting the initial anchors for training the models. The anchors are adjusted using the k-means clustering algorithm to get a number of bounding box dimension clusters of the training dataset equal to the number of anchors needed to define before training the models. Therefore, these anchors will be pairs (x, y) of adequate size for the detection of smoke and fire in images, which will allow a better location of objects of both classes in the test dataset.

The first tests used the default anchors, recommended when proposing the models used, in order to be able to compare the performance of YOLOv3 and YOLOv4. In this test, first, the anchors were adjusted directly to the training dataset and the then test results obtained with the trained models were compared with the results obtained by training with the default anchor values. The test setup was as follows:

- Training with *Fire-Smoke-YOLO* dataset;
- Use of pre-trained networks with *Imagenet* dataset;
- Number of classes = 2 and Channels = 3 (RGB);
- Training up to 6000 epochs;
- Default input size 416 x 416;
- Training of YOLOv4 models;
- Anchor boxes defined as:
  - YOLOv4 default anchors:
    anchors = (12, 16), (19, 36), (40, 28), (36, 75), (76, 55), (72, 146), (142, 110), (192, 243), (459, 401);

  - YOLOv4 anchors adjusted using k-means:
    anchors = (35, 37), (60, 84), (144, 89), (90,160), (200,167), (134,256), (334,188), (233,290), (375,342);
  - YOLOv4 anchors adjusted to **Test images used for training** using k-means:
    anchors = (22, 21), (32, 38), (65, 43), (47, 81), (89,158), (169,131), (289,201), (225,381), (381,286)
  - YOLOv4 anchors adjusted to **Original training dataset with half of the original test dataset** using k-means:
    anchors = (34, 36), (60, 82), (147, 91), (89,160), (200,169), (134,258), (334,190), (235,299), (376,339)

- Testing with half of the *Real-Images-YOLO* dataset;

The tests were carried out, taking into account:

- The use of the original training dataset for training the model;
- The use of half of the original test dataset joined with the original training dataset;
- The use of half of the original test dataset for training;
- The use of 2 training cycles: first with the original training dataset and then with the dataset of test images used for training.

The mAP test results obtained are presented in figure 4.32 as well as in table 4.15.

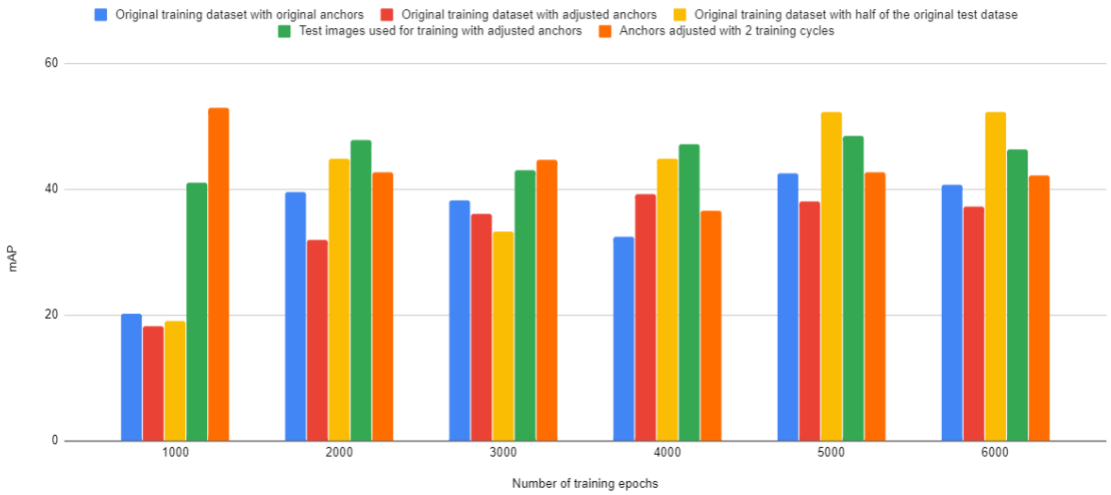


Figure 4.32: mAP test results

The best results were obtained by using two training cycles, first with the original training dataset, followed by a second training cycle using half of the test dataset, and training the model with one cycle, using the original training dataset with half the test dataset.

In figures 4.33 and 4.34, the Average Precision (AP) results obtained with the detections are presented. The test AP results shown in both graphs, in red, refer to the models' second training cycle as well as the mAP test results presented in 4.32 in orange.

When analyzing the graphs together with table 4.15, it can be inferred that the best mAP test results were obtained after training the models with the most significant number of examples of fire and smoke.

	Number of epochs					
Dataset and anchors used	**1000**	**2000**	**3000**	**4000**	**5000**	**6000**
Original training dataset with original anchors	20.11	39.49	38.18	32.46	42.48	40.61
Original training dataset with adjusted anchors	18.16	32.00	36.09	39.11	38.01	37.21
Original training dataset with half of the original test dataset	19.06	44.85	33.28	44.74	**52.24**	52.22
Test images used for training with adjusted anchors	40.99	47.77	43.03	47.15	48.40	46.29
Anchors adjusted with 2 training cycles	**52.98**	42.74	44.71	36.59	42.69	42.14

Table 4.15: mAP results resulting from tests with different training datasets

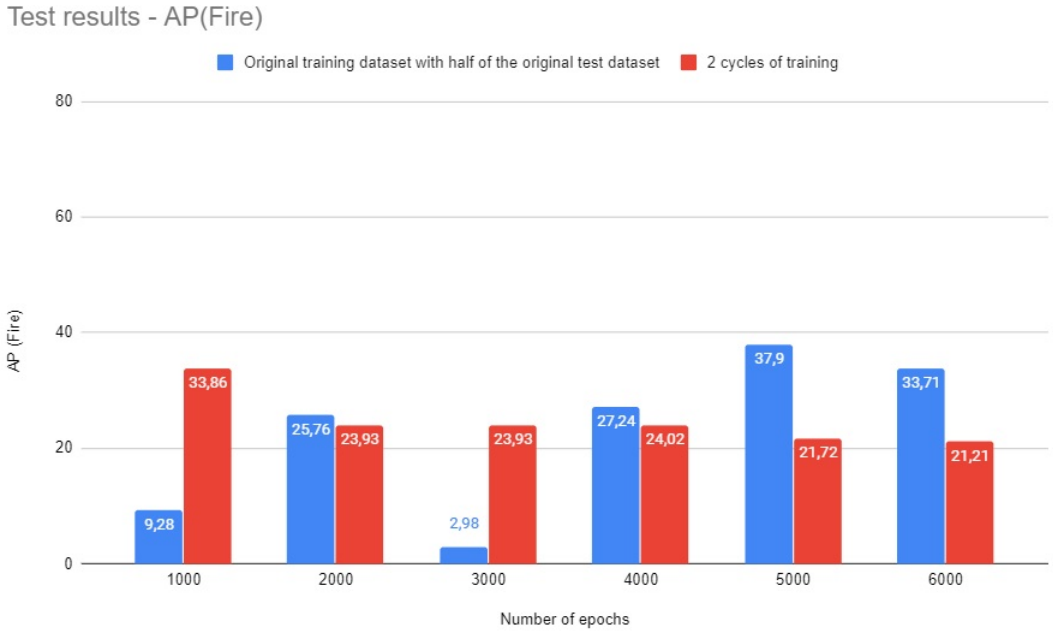


Figure 4.33: Test results AP(Fire)

Figure 4.34: Test results AP(Smoke)

By using two training cycles, using the transfer learning technique twice, the model is first fine-tuned for the detection of objects belonging to the Fire and Smoke classes. The second cycle aims at fine-tuning the model to obtain a better performance in the context of forest fires, and, for this, half of the images belonging to the Real-Images-YOLO dataset are used.

It is possible to conclude that this second training cycle, considering the results of AP (Fire) and AP (Smoke) of graph 4.34, allows the model to detect smoke in the images without so much interference from the background.

The best result obtained in terms of fire detection in the test images was 37.9 AP(Fire), achieved through YOLOv4 training using the original training dataset with half the test dataset, at 5000 epochs. Although this model has better results in detecting fire, the best result for AP(Smoke) was obtained using two training cycles, reaching the value 72.1, at 1000 epochs of the second training cycle.

Tables 4.16 and 4.17 show the best results obtained with this test by training the models with two cycles of training and with the original training dataset joined with half of the original test dataset.

	TP	**FP**	**FN**
Test results - Fire	92	35	146
Test results - Smoke	156	13	57

Table 4.16: Test results obtained with **two cycles of training**

	TP	FP	FN
Test results - Fire	106	47	132
Test results - Smoke	153	49	60

Table 4.17: Test results obtained after training with **Original training dataset with half of the original test dataset**

These tables, 4.16 and 4.17, show that the largest number of detections performed correctly (TP) of objects by the model obtained after training with the original training dataset and half the test dataset.

It can also be observed that both models have a similar number of false negatives, which indicates that they fail equivalently in the detection of several fire spots or smoke columns.

Below are the Precision-Recall curves for the models' performance in detecting objects of the Fire and Smoke classes.

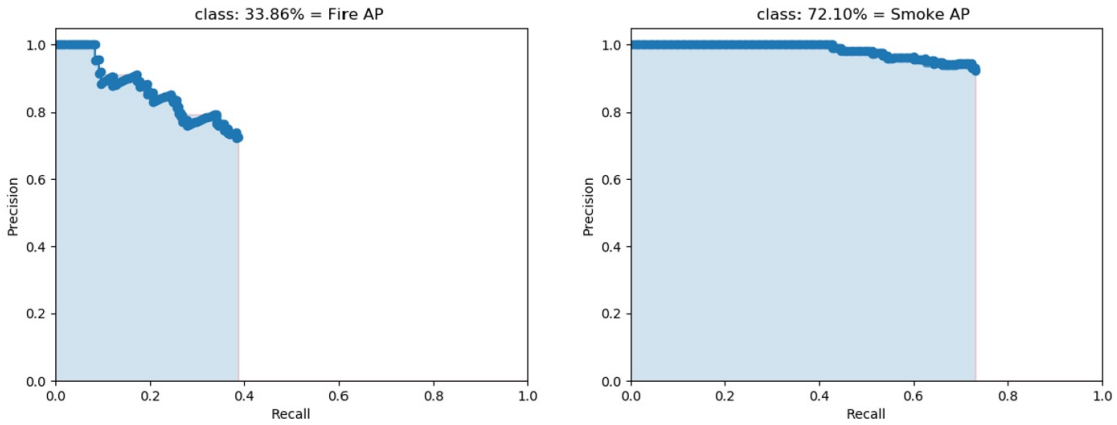


Figure 4.35: Precision Recall curves obtained after **two cycles of training**

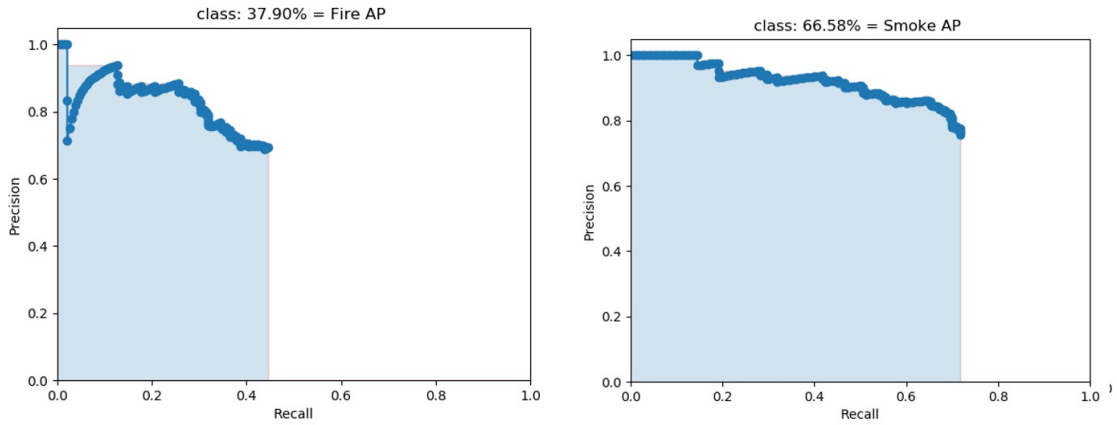


Figure 4.36: Precision Recall curves obtained after training with **Original training dataset with half of the original test dataset**

In figure 4.36 are the results obtained by training with **Original training dataset with half of the original test dataset**, after 5000 epochs and in the figure 4.35 the results obtained using **two training cycles**, after 1000 epochs of the second training cycle.

Making a comparison between the curves resulting from the detection of Smoke objects, on the right in the images, the curve in the image 4.35 is closer to a perfect curve. It is superior to the curve shown in the image 4.36. This difference reflects in the AP(Smoke) values obtained by both trained models.

Although the comparison between the Precision-Recall curves for the detection of objects belonging to the **Smoke** class is possible, the comparison these curves regarding class **Fire** objects, on the left in the images, is more complicated since the detections of both the models result in quite noisy curves. However, it is possible to know that the area under the Precision-Recall curve in figure 4.36 is more extensive, resulting in a higher AP(Fire) value.

As a complementary analysis of the models' performance, the necessary post-processing steps were applied to compare the classification results. The models are compared as to the classification results of the test dataset, described in 4.18 .

Class	**Number of images**
Fire	84
Smoke	94
Neutral	37

Table 4.18: New test dataset description

Table 4.20 presents the classification results obtained after training the model **Original training dataset with half of the original test dataset**, and in table 4.19, the results obtained by the model obtained after **two cycles of training** are shown.

	Predictions		
Ground-truth	**Fire**	**Neutral**	**Smoke**
Fire	68	0	16
Neutral	0	37	0
Smoke	6	20	68

Table 4.19: Confusion matrix results obtained with 2 training cycles

	Predictions		
Ground-truth	**Fire**	**Neutral**	**Smoke**
Fire	60	3	21
Neutral	0	32	5
Smoke	6	11	77

Table 4.20: Confusion matrix results obtained after training the model **Original training dataset with half of the original test dataset**

While the model obtained after training with two cycles, in table 4.19, presents 14 false negatives, the model trained with the large dataset, in table 4.20, presents only 20.

Regarding the number of true positives, the model's performance trained with two cycles surpasses that of the other model, achieving 173 correctly identified images. The model trained with the Original training dataset with half of the original test dataset achieved a total of 169 correctly-classified images.

In conclusion, although the classification results show that the model trained with two cycles is more prone to false negatives, the results show that the addition of a training cycle can be a hypothesis to improve the model's performance.

For a graphical perception of the models' detection results, two examples of images are presented in the examples 4.37 and 4.38.

Figure 4.37: Image prediction example 1

Figure 4.38: Image prediction example 2

The 4.37 image shows the detections made by the models in an image in which only smoke is present. The detections were both performed by the models with a confidence of 96%.

In the image 4.38, the detection results are shown in an image with fire and smoke. On the left, the model trained with two cycles resulted in correct detections of fire and smoke, with the respective degrees of confidence: Smoke: 86%, Fire: 50%, and Fire: 95%.

On the right, the results show an incorrect Fire detection, with 35% confidence. Similar to the other model results, correct detections of fire and smoke are marked, with the respective degrees of confidence: Smoke: 96%, Fire: 48%, and Fire: 72%.

4.2.2 Training and evaluating the model

This section presents the tests performed to optimize the input size parameter used in training and fine-tune the confidence threshold for image fire and smoke detection.

Test for adjusting anchors, considering different input sizes

The test performed regarding the models' training and evaluation for the object detection approach consisted of the variation of the data input size used for training. The results were then evaluated to adjust the performance for the fire and smoke detection problem, varying the confidence threshold.

After optimizing the parameters used for the configuration of the YOLO models and the pre-processing of the images, the dataset composed of the original training dataset with half of the initial test dataset was used. The anchors are adjusted according to the previous test were also used.

The train and test setup can be described as follows:

- Training with *Fire-Smoke-YOLO* dataset;
- Use of pre-trained networks with *Imagenet* dataset;
- Number of classes = 2 and Channels = 3 (RGB);
- Training up to 6000 epochs;
- Tested input sizes: 416 x 416 (default and recomended input size), 608 x 608 (recomended value for increasing the input size) and 224 x 224 (decreasing the input size to the one used in YOLOv2);
- Training of YOLOv4 model;
- Use of YOLOv4 anchors adjusted to **Original training dataset with half of the original test dataset** using k-means:

  anchors = (34, 36), (60, 82), (147, 91), (89,160), (200,169), (134,258), (334,190), (235,299), (376,339)
- Testing with half of the *Real-Images-YOLO* dataset;

The best mAP test results were 52.22% obtained using model YOLOv4 trained with input size of 416 by 416, after 6000 training epochs, as shown in graph 4.39. These results correspond to 33.71% AP(Fire) and 70.73% AP(Smoke) results, as shown in graphs 4.40 and 4.41.

However, although the model trained considering input size 416 by 416 obtains better performance in the detection of fire and smoke, the test results of the model using input size 224 by 224 are quite close. In some of the training's intermediate points, the mAP test results obtained by this second model exceed the results obtained by the first.

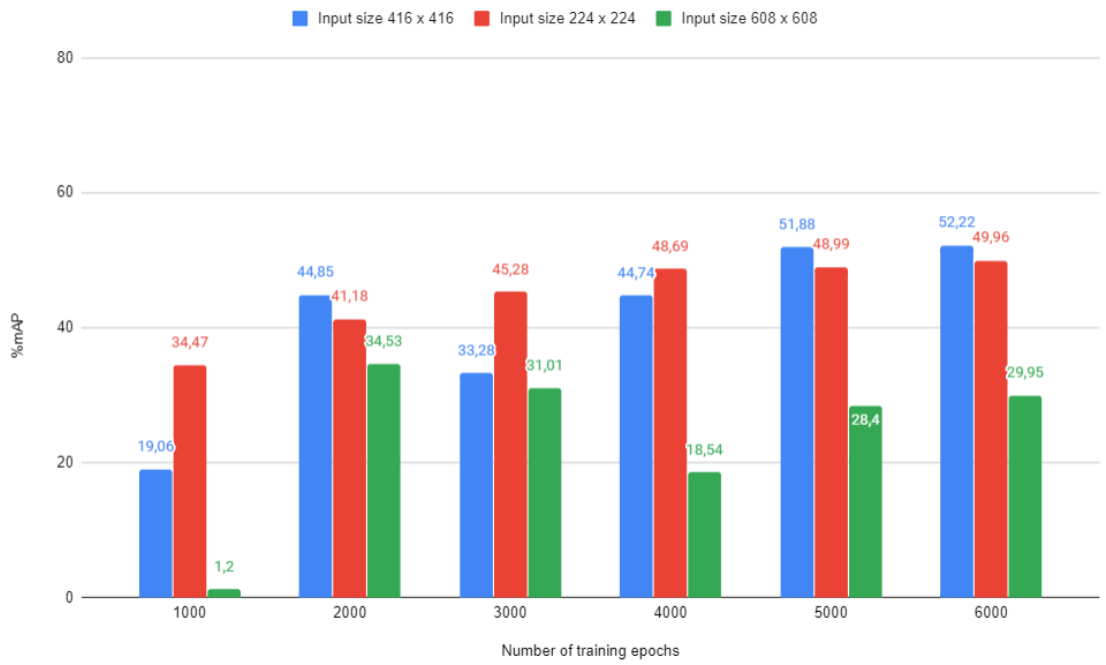


Figure 4.39: mAP test results obtained with variation

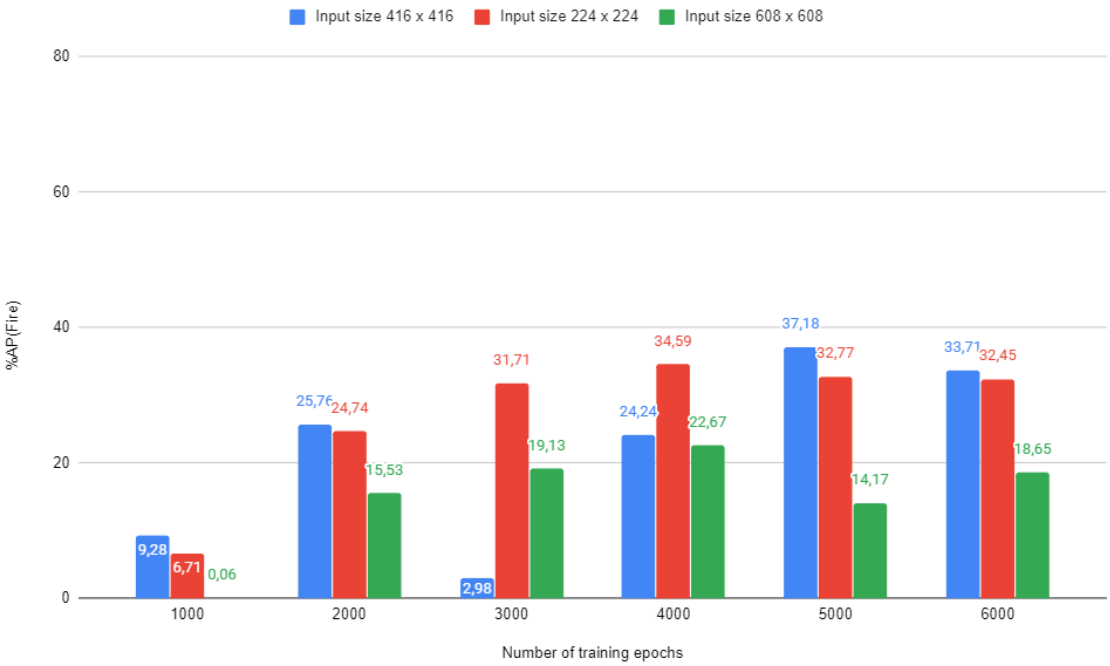


Figure 4.40: AP(Fire) test results obtained with variation

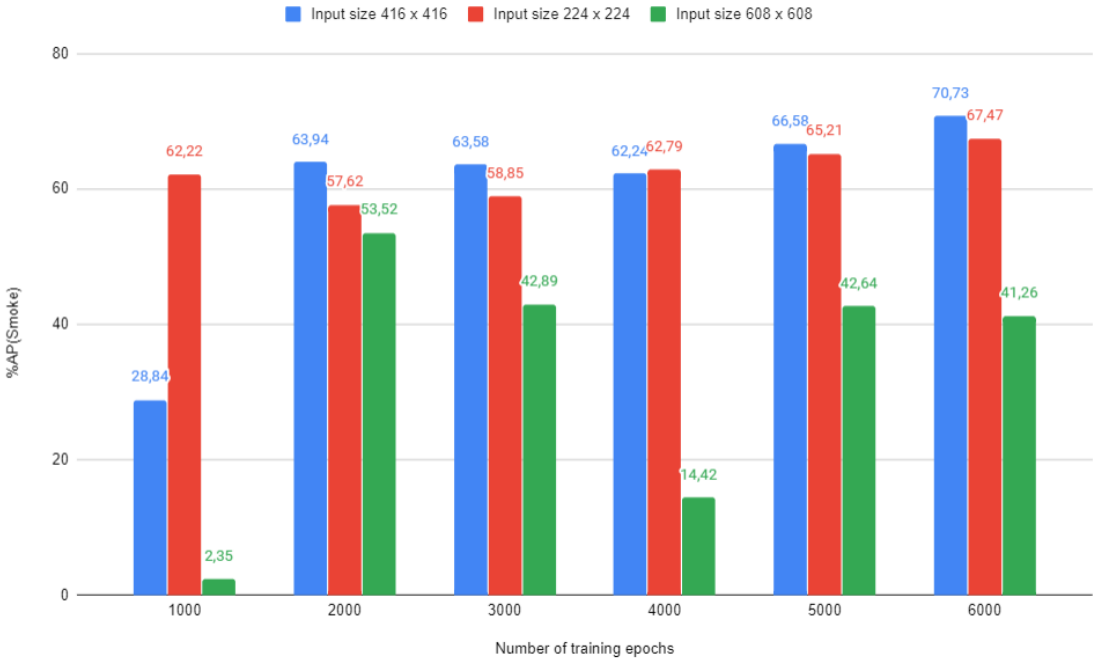


Figure 4.41: AP(Smoke) test results obtained with variation

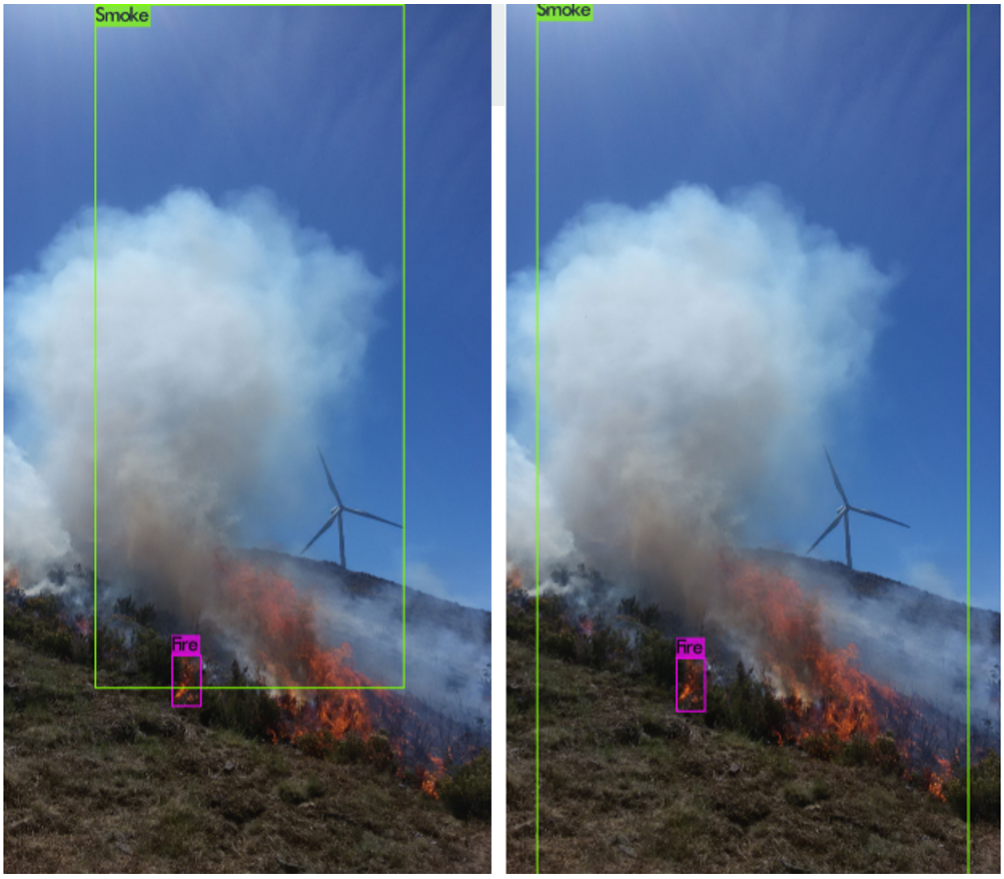
As a complementary analysis of the results and for a visual comparison of the images' detection results, two distinct detection examples are presented in figures 4.42 and 4.43.

In figure 4.42, example 1 represents the detections made by the models in an image that contains smoke and fire, considering the models trained with input sizes 416 by 416, and 608 by 608. Graphically the results obtained by the model trained with input size 416 by 416 are similar to the results obtained after training with 608 by 608. Both classify the smoke column present in the image correctly, differing in confidence in the detection. Using input size 608 by 608, the Smoke confidence score is 77%, and with 416 by 416, the confidence score is lower, reaching only 28 %. The other model, trained with input size set to 224 x 224, does not detect any of the objects present in the image.

In example 2, presented in figure 4.43 , the results of detections obtained in another image in which smoke and fire are also present are presented. In this case, the models' performance when identifying smoke is similar in the three trained models. The results obtained by models trained with input size 224 x 224 and 416 by 416 are similar for fire detection. However, while the model trained with input size 224 by 224 identifies smoke with a confidence score of 70%, the other does it with 100%.

As for fire, both have repeated detections, whereas, while the model trained with input size 416 by 416 has a maximum confidence score of 91%, the one trained with the input size set to 224 x 224 only reaches 65%. The other model performs the detections with the maximum confidence scores of 77% and 26%.

Although they present repeated detections, the models trained with input sizes set to 224 by 224 and 416 by 416 have better performance in detecting fire and smoke in images, of which the model trained with the input size set to 416 by 416 stands out. For that reason, the model chosen to perform the test to adjust the confidence threshold, presented below, was the one trained with input size 416 by 416.



Results with **input size 416 by 416** Results with **input size 608 by 608**

Figure 4.42: Test results obtained using different input sizes, example 1

Results obtained with **input size 416 x 416** Results obtained with **input size 608 x 608**

Figure 4.43: Test results obtained using different input sizes, example 2

Testing the best model for confidence threshold adjustment

This test aimed to adjust the confidence threshold to be used for the operation of the system. For a **confidence threshold T**, all detections for which the **confidence score C** is below the defined threshold will be discarded. That is, only if **C**> **T** that the object is considered to be detected by the model.

As such, the definition of this threshold will affect the number of detections made by the models and, with this test, it was intended to ascertain the effects of varying the confidence threshold on the results of mAP, AP (Fire), AP (Smoke) but also on the results classification of images. Prior to this test, the confidence threshold value in the studies YOLOv3 [57] and YOLOv4 [22] was used.

In order to adjust the image evaluation parameters, in addition to the metrics used for the previous tests, the models are also evaluated for classification performance. More specifically, this way of evaluating performance means that each model is analyzed for the ability to correctly detect the location of fire and smoke in the images (detection capability) and for the ability to classify the test images (classification capability) correctly.

The testing phase consisted of:

- Use the best model obtain in the previous test, trained with input size 416 by 416, trained up to 6000 epochs;
- Confidence thresholds considered: 5%, 10%, 15%, 20% and 25%;
- Testing with half of the *Real-Images-YOLO* dataset;

The effect of the variation of the confidence threshold can be seen in the graph of figure 4.44, which presents the mAP results obtained considering the confidence thresholds 5%, 10%, 15%, 20% and 25%, evaluating the model resulting from the previous test.

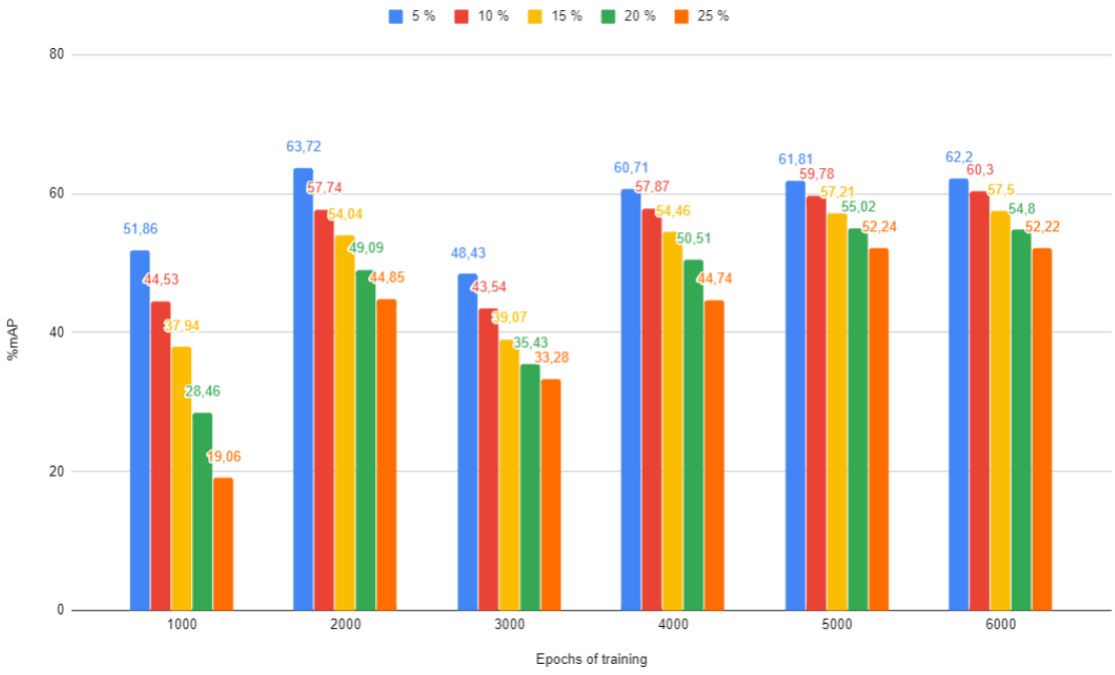


Figure 4.44: Results obtained considering varying confidence threshold

From the results in the graph of figure 4.44, it can be confirmed that the models, with the lowering of the confidence threshold, obtain worse results of test mAP. This means the highest number of valid detections (where C> T) that is carried out is related to correct detections, and the number of false negatives decreases. That is, more fire spots and more smoke columns are detected in the images.

Therefore, a more detailed analysis was made regarding two operating points: with **confidence thresholds 5 and 15%**. For this reason, it was considered the model that obtained the best results across all thresholds considered, trained up to 6000 epochs.

Regarding this model, the graphs of the figures 4.45 and 4.46 show the Precision-Recall curves obtained for confidence 15% and for 5%.

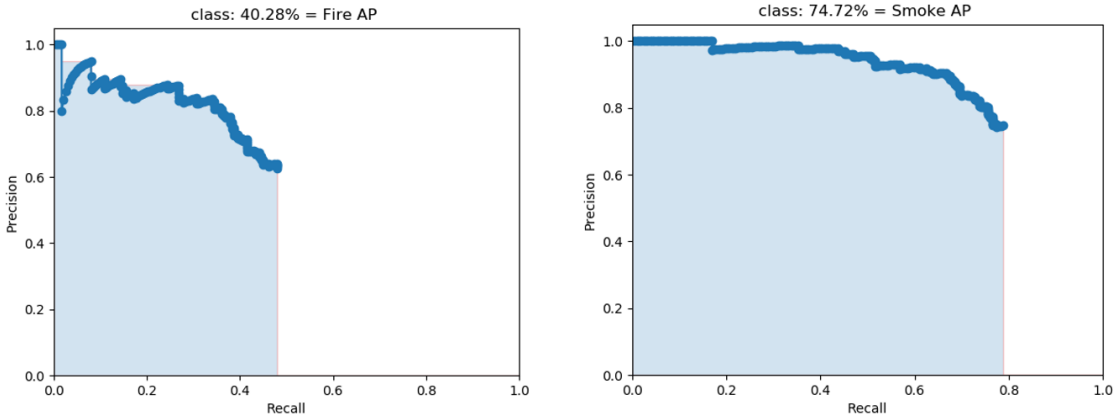


Figure 4.45: Precision - Recall curves obtained for confidence 15%

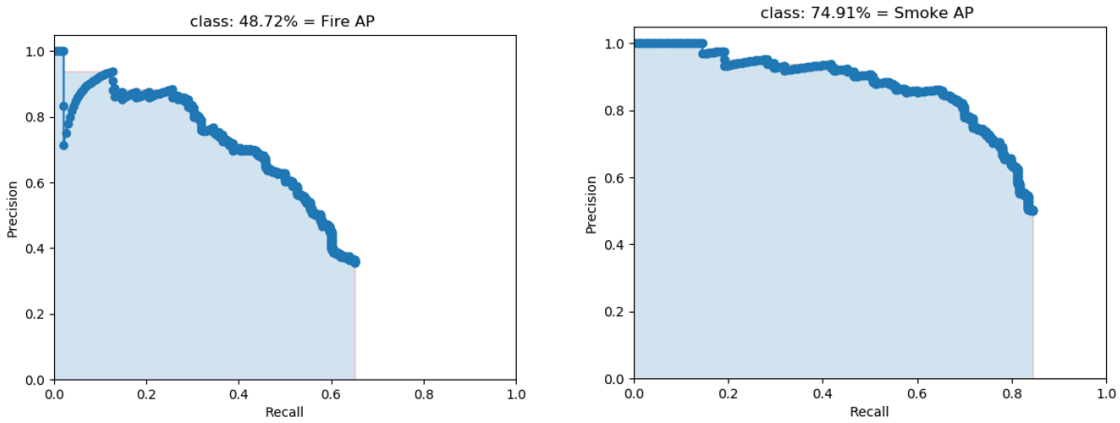


Figure 4.46: Precision - Recall curves obtained for confidence 5%

By complementing the analysis of graph 4.44 with the Precision-Recall curves, it can be noticed that the improvement in the mAP results is mainly due to the improvement in the fire detection results in the images. With the increase of the confidence threshold, both curves show a decrease in the Recall values, and the Precision values show a more significant oscillation up and down (which is more noticeable in the graphs related to AP(Fire)).

The AP(Smoke) Precision-Recall curves, on the right in the images, also show some observable differences. The curve obtained setting the confidence threshold at 5% is more extensive, which is reflected in a slight increase in the AP(Smoke) result. However, the test curve where the threshold was set at 15% is higher, which indicates that it will detect smoke objects more accurately.

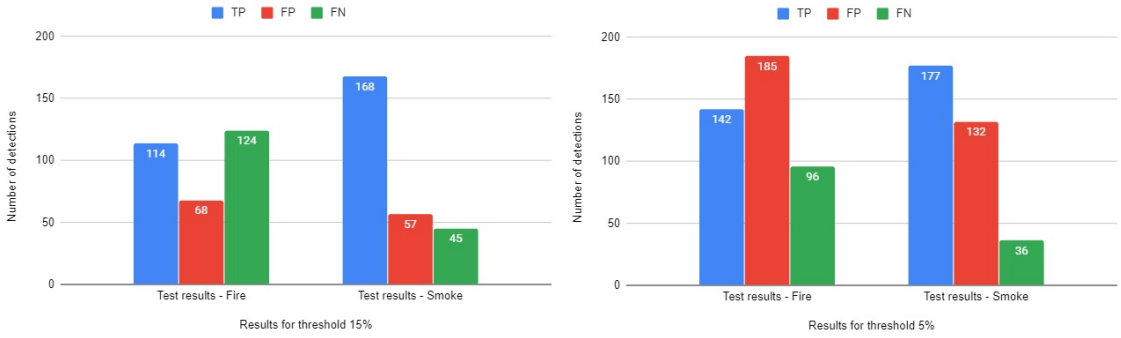


Figure 4.47: Test results comparison

The curves' analysis is complemented by comparing the total of True Positives, False Positives, and False Negatives detections obtained by each of the models presented in figure 4.47.

With the lowering of the confidence threshold used in the test, the False Positive values show a considerable increase. However, considering the mAP values analyzed previously, it can be concluded that most of these are due to location errors, that is, repeated detections of the same object in the images.

This assumption is based on the simultaneous increase in the value of True Positive detections and a decrease in the value of False Negatives. A lower value of False Negatives in the detections means that, of the total number of smoke columns and fires present in the images, a smaller number of objects remain undetected.

In order to confirm the effect of this variation on the image classification results, the results of the confusion matrices were also analyzed after applying the step of post-processing the results. These matrices are presented in tables 4.21. 4.22, for confidence threshold of 15% and 4.23. 4.24, for confidence threshold of 5%.

	Predictions – confidence 15 %		
Ground-truth	**Fire**	**Neutral**	**Smoke**
Fire	67	0	17
Neutral	0	31	6
Smoke	6	5	83

Table 4.21: Confusion matrix with confidence threshold 15% considering 3 classes

	Predictions – confidence 15 %	
Ground-truth	**Fire/Smoke**	**Neutral**
Fire/Smoke	173	5
Neutral	6	31

Table 4.22: Confusion matrix with confidence threshold 15% considering 2 classes

	Predictions – confidence 15 %		
Ground-truth	**Fire**	**Neutral**	**Smoke**
Fire	78	0	6
Neutral	1	22	14
Smoke	14	0	80

Table 4.23: Confusion matrix with confidence threshold 5% considering 3 classes

	Predictions – confidence 5 %	
Ground-truth	**Fire/Smoke**	**Neutral**
Fire/Smoke	178	0
Neutral	15	22

Table 4.24: Confusion matrix with confidence threshold 5% considering 2 classes

In contrast to the test performed considering a confidence threshold of 15%, when analyzing the confusion matrices of the image classification results, it can be seen that the test with the threshold set to 5% was able to obtain 0 False Negatives. This result means that none of the images that contained smoke or fire were classified as neutral.

However, as expected, the number of false positives increased from 6 to 15 images when using a lower confidence threshold. As such, in this respect, the use of a 15% confidence threshold in the FireLoc application has advantages, managing to eliminate a more significant number of crowdsourced images that do not portray forest fire situations.

As representatives of the observable differences in practice, some images with fire and smoke detections are presented in figures 4.48 and 4.49.

Figure 4.48: Test results image detection comparison, example 1

Figure 4.49: Test results image detection comparison, example 2

In figure 4.48, example 1 represents the detections made by the models in an image with smoke, considering the confidence thresholds of both the proposed operating points: 5% and 15%. In this example, the performance of the models in detecting the smoke columns is similar. However, when the 5% threshold is used, on the left in the image, there is a repeated smoke detection (which would be considered a False Positive when calculating the AP value (Smoke) and, consequently, in calculating the mAP value).

In example 2, presented in figure 4.49, the results of detections obtained in an image in which smoke and fire are present are shown. In this case, the performance of the models when identifying smoke is also similar. As for fire detection, the model presents repeated fire detections, which would also be considered False Positives in the metrics for assessing the detection performance. These results also indicate that the tested model has greater confidence in the detection of smoke than in the detection of fire in the images.

4.2.3 Discussion of Detection Results

Using the object detection approach in images, a better distinction between fire and smoke present in the images was possible. Compared to the classification approach, using the YOLO models has the advantage of making it possible to obtain information about fire and smoke location in the images' space.

The YOLO models used in this approach are deeper, and, as such, large amounts of data are needed to obtain good results in detecting objects in images. Similarly to what was done for the previous approach, the transfer learning technique was used.

The lack of specific datasets led to the creation of new training and test datasets, from the images used in the classification approach, annotated manually. In this approach, the IoU was adjusted for the networks used (instead of the typical value 0.5), lowering it to 0.3, which allowed avoiding the consideration of location errors in the evaluation of the detection results.

The tests carried out as part of the development of this approach are summarized in table 4.25. First, tests were performed to choose the model to be used to optimize the considered parameters. Then, tests were performed with YOLOv4 to improve the models' detection and classification performance, adjusting the anchors, the input size, and the confidence threshold to be used.

Test name	Test performed	Test objective	Results evaluation metrics
Initial tests	Performance comparison between the models considered in this approach (YOLOv3 and YOLOv4)	Choose the model to be used in the following tests	mAP and AP per class
Test performed with anchor adjustment	Performance comparison between the best model (YOLOv4) considering original anchors and anchors adjusted to the training dataset	Adjust anchors to obtain better object detection	
Tests for adjusting anchors, considering different training datasets	Performance comparison between the best model (YOLOv4) considering original anchors and anchors adjusted to the training dataset, varying the training dataset used		mAP and AP per class, Precision-Recall curves, confusion matrices considering 3 and 2 classes, visual comparison of image detections
Test for adjusting anchors, considering different input sizes	Performance comparison between the best model (YOLOv4) with best results in previous test, trained with varying image input sizes	Adjust anchors to obtain better object detection and image classification	mAP and AP per class, Precision-Recall curves, confusion matrices considering 3 and 2 classes, visual comparison of image detections
Testing the best model for confidence threshold adjustment	Performance comparison between results obtained with different confidence thresholds, using the model with best performance in the previous tests	Optimize confidence threshold for better object detection and image classification	mAP and AP per class, Precision-Recall curves, confusion matrices considering 3 and 2 classes, visual comparison of image detections

Table 4.25: Tests performed for image object detection approach

The anchors used in the training of the models were optimized as well as the input size of the images. Finally, the confidence threshold to be used for the evaluation of the images was optimized. As a result of this test, 2 points for system operation are proposed, using confidence thresholds 5% and 15%.

The proposal of 2 operating points will allow adjusting the system's performance, making it more permeable to false positives (reducing the threshold used to evaluate the images to 5 %). As an alternative, in times when there is an increase in the number of reported fires, create a more significant restriction against false positives (increasing the threshold to 15 %).

It was also possible to observe that the use of new training cycles can improve the models' performance. This result is interesting since it allows future improvement, with the possibility of using new images available with forest fire situations to fine-tune the proposed YOLOv4 model.

This system will allow the detection and location of the fire and smoke present in the images, and the classification results for each image when applying the proposed post-processing steps.

Although not within the work scope, YOLO models still manage to detect objects on video, which can be useful if the FireLoc application starts to accept this type of submissions.

This page is intentionally left blank.

Chapter 5

Conclusions and Future Work

Efficient fire reporting systems are essential as they allow for a quicker and more effective fire fighting media response. One possibility for the implementation of these systems is the use of crowd reporting, which allows people to present at the locations to identify and report the situation almost immediately, such as FireLoc. For these systems to be used in real contexts, it is necessary to process a large number of images submitted simultaneously. Therefore, it is essential to have a system that serves as a filter to check if each submitted image contains signs of a forest fire.

In this thesis, an intelligent system for the detection of smoke and fire in static images was developed for integration in the FireLoc project as a tool for evaluating the image contributions made by users. In this chapter, the final product and its development are analyzed, as well as the future work and the difficulties encountered.

Based on the state-of-the-art analysis, the advantages of using deep learning to solve image recognition problems become clear. Despite the need for a large number of images to train artificial neural networks, using techniques such as transfer learning and data augmentation, it is possible to obtain a reliable model for detecting fire and smoke in images, as demonstrated by the existing systems studied.

For the recognition of forest fire indications in static images, two approaches were studied: image classification and image object detection. Initially, an image classification system was implemented, using state-of-the-art ResNet models. Although it was possible to improve the results by optimizing some training parameters of the models and dataset augmentation, it presented some confusion in identifying fire and smoke.

Therefore, it was considered an alternative approach of detecting objects in images, considering fire and smoke the objects to be detected. This approach allowed not only the distinction between the two but also the location in the image space of the objects found. With this approach, the differences between fire and smoke in images by the models become more accurate, allowing, in the future, to identify the location of the reported fire from the point of view of the users who send the images.

Based on the tests performed, it is then proposed to recognize fire and smoke and static images using an image object detection approach. After considering the analysis of the results, the use of a trained YOLOv4 as a screening tool for the contributions of the users of the FireLoc application is proposed since it presents good results in the test performed with images collected in the context of a forest fire.

The main difficulties encountered were the lack of annotated data for training the models

and the consequent subjectivity of the manual annotation of the datasets. Another aspect that hindered the work carried out was the long time required for training the models, especially the more complex ones used in the object detection approach, the YOLO models.

The following contributions resulted from the work developed with the proposed objectives in mind:

- The documentation of the state-of-the-art regarding fire recognition methodologies, which can be used for the development of fire and smoke recognition systems in static images or video, presented in chapter 2;
- The implementation of a fire and smoke detection system in images to be integrated into the FireLoc system;
- The proposal for specific image datasets for training and testing of YOLO networks, which can be used for the development of smoke and fire detection systems in static images or videos;
- A system test on the server where it will be used, simulating a real image submission situation, evaluating the results regarding the fire and smoke detection performance in each image, and regarding the classification performance, considering the images as a whole, proposing two operating points for the system.

The system has, however, room for improvement. Once the application is put into operation, with the users' submitted data, it will become possible, for example, to improve the performance of the model, training with another cycle using the submitted images, thus enhancing its generalization capacity. It will also be possible to compare the two operating points and confirm the differences observed between them in the test performed.

Due to the recent interest in the image and video object recognition scientific area, the performances of new models that can be developed can also be compared. Examples of these models are PP-YOLO [47] (released in August 2020) or even YOLOv5 [39] (released at the end of May 2020, but for which, at the time of writing this dissertation, there is no peer-reviewed paper for).

These can be tested and compared to the trained YOLOv4, using the proposed annotated datasets for the training and testing phases and according to the image object detection approach presented in chapter 2. The proposed dataset can also be used to develop other systems since the annotation used is in the YOLO annotation format.

This page is intentionally left blank.

References

[1] AML - Advanced Mobile Location. https://www.sg.mai.gov.pt/Noticias/Paginas/Demonstra%C3%A7%C3%A3o-real-do-AML---Advanced-Mobile-Location.aspx. accessed 2020-01-17.

[2] Australia's Deadly Wildfires in Photos: The View from Space. https://www.space.com/australia-wildfires-satellite-images-2019-2020.html. accessed 2020-01-19.

[3] Bushfire-destroyed homes should not be rebuilt in riskiest areas, experts say. https://www.theguardian.com/australia-news/2020/jan/19/bushfire-destroyed-homes-should-not-be-rebuilt-in-riskiest-areas-experts-say. accessed 2020-01-19.

[4] Classification report. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html. accessed 2020-08-10.

[5] COCO Dataset - Detection Evaluation. https://cocodataset.org/#detection-eval. accessed 2020-08-10.

[6] Confusion matrix. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html. accessed 2020-08-10.

[7] Convolutional Neural Networks (CNNs / ConvNets). http://cs231n.github.io/convolutional-networks/. accessed 2020-01-17.

[8] Fatalities due to the most significant wildfires worldwide as of 2016. https://www.statista.com/statistics/234723/fatalities-due-to-the-most-significant-wildfires/. Accessed: 2020-01-10.

[9] Fire-Smoke-Dataset. https://github.com/DeepQuestAI/Fire-Smoke-Dataset/releases/download/v1/FIRE-SMOKE-DATASET.zip. accessed 2020-08-22.

[10] FireLoc - Localize o Fogo. https://fireloc.org/. accessed 2020-09-06.

[11] ImageNet. http://www.image-net.org/. accessed 2020-08-22.

[12] Incêndios. Portugal tem 41.017 hectares de área ardida em 2019. https://expresso.pt/sociedade/2019-10-01-Incendios.-Portugal-tem-41.017-hectares-de-area-ardida-em-2019. accessed 2020-01-19.

[13] An intuitive guide to convolutional neural networks. https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/. Accessed: 2020-01-17.

[14] LabelImg. https://github.com/tzutalin/labelImg#labelimg. accessed 2020-05-25.

[15] Project RESCUER – new communication platform to save lives. http://www.rescuer-project.org/. accessed 27-12-2019.

[16] TORCHVISION.MODELS. https://pytorch.org/docs/stable/torchvision/models.html. accessed 2020-01-18.

[17] YOLO: Real-Time Object Detection - Training YOLO. https://pjreddie.com/darknet/yolov1/. accessed 2020-08-22.

[18] Yolo v4, v3 and v2 for Windows and Linux. https://github.com/AlexeyAB/darknet. accessed 2020-08-10.

[19] Octavio Arriaga, Paul Plöger, and Matias Valdenegro-Toro. Image Captioning and Classification of Dangerous Situations. nov 2017.

[20] Marcos V.N. Bedo, Gustavo Blanco, Willian D. Oliveira, Mirela T. Cazzolato, Alceu F. Costa, Jose F. Rodrigues, Agma J.M. Traina, and Caetano Traina. Techniques for effective and efficient fire detection from social media images. *ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings*, 1(June):34–45, 2015.

[21] Sam G Benjamin. A Comparative Analysis on Different Image Processing Techniques for Forest Fire Detection. *IJCSN International Journal of Computer Science and Network*, 5(1), 2016.

[22] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. (May), 2020.

[23] J. Cartucho, R. Ventura, and M. Veloso. Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2336–2341, 2018.

[24] T. Celik, H. Demirel, and H. Ozkaramanli. Automatic fire detection in video sequences. *European Signal Processing Conference*, (January), 2006.

[25] T. Celik and Kai-Kuang Ma. Computer Vision Based Fire Detection in Color Images. pages 258–263, 2008.

[26] Sayantan Chatterjee, Faheem H Zunjani, Souvik Sen, and Gora C Nandi. Real-Time Object Detection and Recognition on Low-Compute Humanoid Robots using Deep Learning.

[27] Thou-ho Chao-ho Chen, Ping-hsueh Wu, and Yung-chuen Chiou. An Early Fire-Detection Method Based on Image Processing. pages 1707–1710, 2004.

[28] Daniel Y. T. Chino, Letricia P. S. Avalhais, Jose F. Rodrigues, and Agma J. M. Traina. BoWFire: Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis. jun 2015.

[29] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[30] Sebastien Frizzi, Rabeb Kaabi, Moez Bouchouicha, Jean Marc Ginoux, Eric Moreau, and Farhat Fnaiech. Convolutional neural network for video fire and smoke detection. *IECON Proceedings (Industrial Electronics Conference)*, pages 877–882, 2016.

[31] Ross Girshick. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1440–1448, 2015.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[33] Oluwarotimi Giwa and Abdsamad Benkrid. Fire detection in a still image using colour information. Technical report.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. dec 2015.

[35] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7574 LNCS(PART 3):340–353, 2012.

[36] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017.

[37] Arpit Jadon, Mohd. Omama, Akshay Varshney, Mohammad Samar Ansari, and Rishabh Sharma. FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications. may 2019.

[38] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7(3):128837–128868, 2019.

[39] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, lorenzomammana, tkianai, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, Jake Poznanski, Lijun Yu , changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, hopesala, pritul dave, and yzchen. ultralytics/yolov5: v3.0, August 2020.

[40] Byoungjun Kim and Joonwhoan Lee. A Video-Based Fire Detection Using Deep Learning Models. *Applied Sciences*, 9(14):2862, 2019.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[42] D Krstinić, D Stipaničev, and T Jakovčević. Histogram-based smoke segmentation in forest fire detection system. *Information Technology and Control*, 38(3):237–244, 2009.

[43] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[44] C. Li and Y. Bai. Fire flame image detection based on transfer learning. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 370–374, Nov 2018.

[45] Pu Li and Wangda Zhao. Image fire detection algorithms based on convolutional neural networks. *Case Studies in Thermal Engineering*, 19(March), 2020.

[46] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

[47] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, and Qingqing Dang. PP-YOLO: An Effective and Efficient Implementation of Object Detector.

[48] Niall O Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep Learning vs . Traditional Computer Vision. (Cv).

[49] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. chapter 8, pages 158–162. Cambridge University Press, Cambridge, UK, 2008.

[50] Wentao Mao, Wenpeng Wang, Zhi Dou, and Yuan Li. Fire Recognition Based On Multi-Channel Convolutional Neural Network. *Fire Technology*, 54(2):531–554, mar 2018.

[51] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *ICML*, pages 807–814. Omnipress, 2010.

[52] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[53] Yan Qiang, Bo Pei, and Juanjuan Zhao. Forest Fire Image Intelligent Recognition based on the Neural Network. *Journal of Multimedia*, 9(3), 2014.

[54] Param S. Rajpura, Hristo Bojinov, and Ravi S. Hegde. Object Detection Using Deep CNNs Trained on Synthetic Images. jun 2017.

[55] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:779–788, 2016.

[56] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:6517–6525, 2017.

[57] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. 2018.

[58] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[59] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of Degradations on Deep Neural Network Architectures. pages 1–11, 2018.

[60] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[62] Lei Sun. ResNet on Tiny ImageNet. Technical report.

[63] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. Technical report, 2013.

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. sep 2014.

[65] Wonjae Lee, Seonghyun Kim, Yong-Tae Lee, Hyun-Woo Lee, and Min Choi. Deep neural networks for wild fire detection with unmanned aerial vehicle. pages 252–253, Jan 2017.

[66] Tong Yu and Hong Zhu. Hyper-Parameter Optimization: A Review of Algorithms and Applications. pages 1–56, 2020.

[67] Qi Xing Zhang, Gao Hua Lin, Yong Ming Zhang, Gao Xu, and Jin Jun Wang. Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images. In *Procedia Engineering*, volume 211, 2018.

[68] Jianhui Zhao, Zhong Zhang, Shizhong Han, Chengzhang Qu, Zhiyong Yuan, and Dengyi Zhang. SVM based forest fire detection using static and dynamic features. *Computer Science and Information Systems*, 8(3):821–841, 2011.

[69] Zhong Qiu Zhao, Peng Zheng, Shou Tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

This page is intentionally left blank.

Appendices

Appendix A

Images from *Fire-Smoke-Dataset*

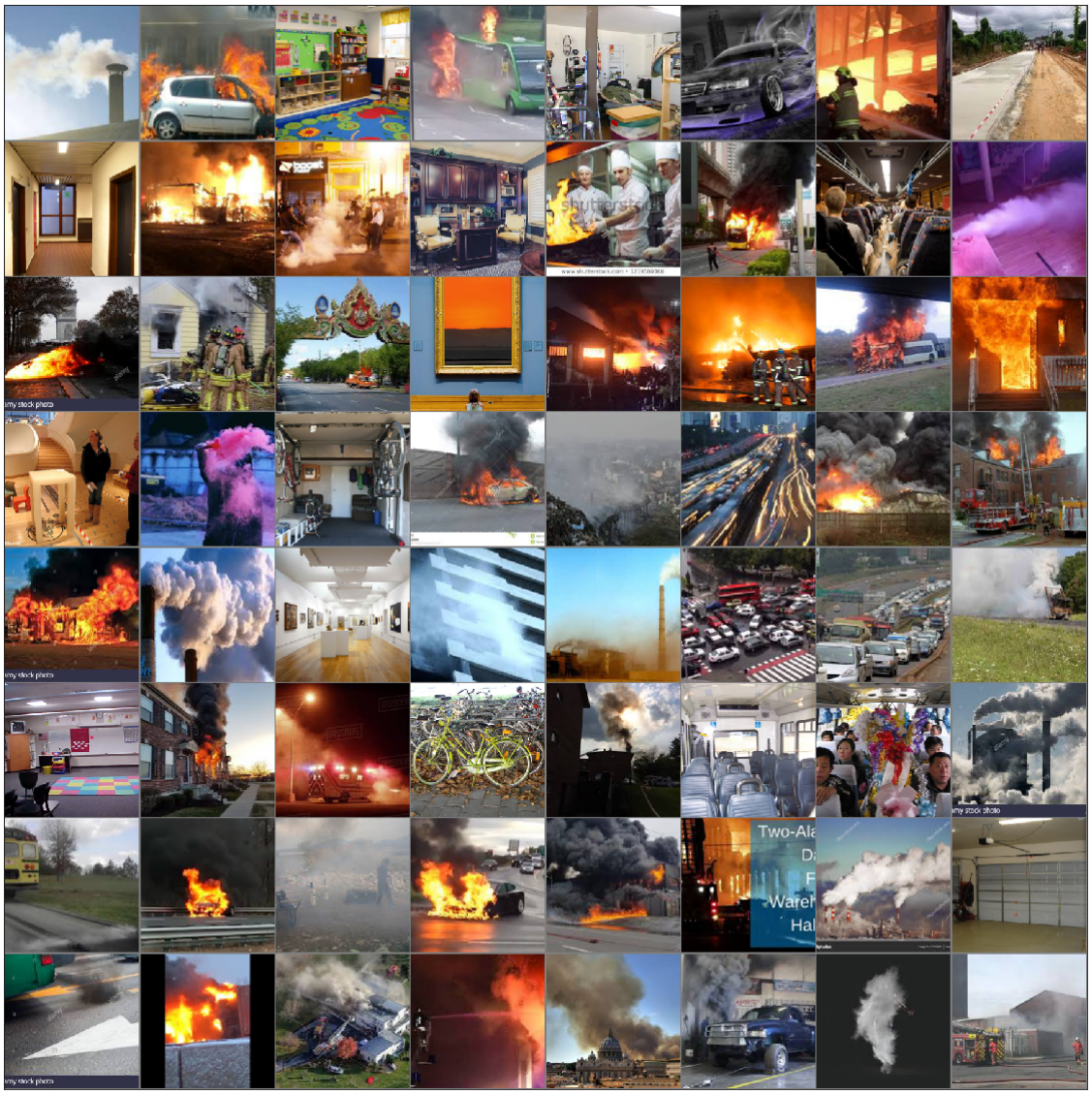


Figure 1: Example Fire-Smoke-Dataset images

Images images from *Real-Images-Dataset*

Figure 2: Example images from *Fire-Smoke-Dataset*

This page is intentionally left blank.

Appendix B

Test Results for Image classification approach

	Number of training epochs									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
w1	0.65	0.71	0.73	0.72	0.73	0.74	0.73	0.75	0.73	0.75
w2	0.69	0.69	0.73	0.74	0.74	0.73	0.72	0.73	0.72	0.73
w3	0.65	0.7	0.73	0.75	0.72	0.73	0.74	0.72	0.72	0.73
w4	0.63	0.69	0.73	0.73	0.73	0.75	0.73	0.72	0.73	0.72
w5	0.67	0.69	0.72	0.71	0.7	0.72	0.7	0.75	0.75	0.73
w6	0.61	0.67	0.7	0.72	0.74	0.74	0.74	0.73	0.75	0.73
w7	0.65	0.73	0.71	0.72	0.71	0.71	0.73	0.75	0.72	0.75
w8	0.57	0.67	0.7	0.71	0.72	0.74	0.72	0.74	0.72	0.72
w9	0.64	0.69	0.74	0.74	0.73	0.72	0.73	0.75	0.73	0.72
w10	0.71	0.72	0.73	0.76	0.72	0.74	0.73	0.75	0.73	0.73

Table 1: Test results obtained with ResNet 50 using data augmentation in all 10 runs, considering 3 classes every 50 epochs (up to 500 epochs)

	Number of training epochs									
	550	**600**	**650**	**700**	**750**	**800**	**850**	**900**	**950**	**1000**
w1	0.74	0.73	0.75	0.73	0.76	0.76	0.74	0.74	0.71	0.73
w2	0.74	0.73	0.75	0.74	0.75	0.73	0.72	0.74	0.73	0.74
w3	0.74	0.72	0.72	0.74	0.74	0.75	0.75	0.75	0.73	0.74
w4	0.74	0.75	0.74	0.76	0.75	0.76	0.74	0.74	0.74	0.73
w5	0.74	0.73	0.74	0.75	0.74	0.74	0.75	0.75	0.72	0.74
w6	0.71	0.74	0.72	0.73	0.74	0.75	0.75	0.74	0.74	0.76
w7	0.73	0.73	0.72	0.75	0.72	0.74	0.73	0.74	0.74	0.73
w8	0.76	0.75	0.74	0.74	0.74	0.76	0.74	0.74	0.74	0.74
w9	0.74	0.74	0.73	0.74	0.74	0.73	0.74	0.75	0.74	0.74
w10	**0.76**	0.75	0.74	0.73	0.73	0.75	0.74	0.74	0.74	0.73

Table 2: Test results obtained with ResNet 50 using data augmentation in all 10 runs, considering 3 classes every 50 epochs

	Number of training epochs									
	50	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**
w1	0.77	0.83	0.86	0.86	0.85	0.86	0.86	0.87	0.85	0.85
w2	0.79	0.84	0.84	0.85	0.86	0.84	0.84	0.85	0.85	0.86
w3	0.76	0.85	0.85	0.87	0.85	0.85	0.86	0.84	0.84	0.85
w4	0.77	0.81	0.85	0.85	0.85	0.87	0.85	0.84	0.85	0.85
w5	0.80	0.82	0.84	0.83	0.84	0.85	0.85	0.87	0.85	0.85
w6	0.74	0.81	0.83	0.85	0.86	0.87	0.87	0.87	0.88	0.86
w7	0.78	0.83	0.83	0.84	0.83	0.83	0.85	0.86	0.84	0.85
w8	0.72	0.8	0.83	0.84	0.84	0.87	0.85	0.86	0.85	0.85
w9	0.77	0.82	0.84	0.86	0.86	0.86	0.85	0.86	0.85	0.84
w10	0.82	0.85	0.85	0.87	0.85	0.86	0.86	0.87	0.86	0.85

Table 3: Test results obtained with ResNet 50 using data augmentation in all 10 runs, considering 2 classes every 50 epochs (up to 550)

	Number of training epochs									
	550	**600**	**650**	**700**	**750**	**800**	**850**	**900**	**950**	**1000**
w1	0.85	0.86	0.86	0.86	0.88	0.87	0.85	0.84	0.84	0.84
w2	0.86	0.86	0.86	0.86	0.86	0.82	0.84	0.86	0.84	0.85
w3	0.86	0.84	0.84	0.85	0.85	0.85	0.85	0.85	0.85	0.85
w4	0.86	0.86	0.85	0.87	0.86	0.86	0.85	0.84	0.84	0.84
w5	0.86	0.86	0.86	0.85	0.85	0.84	0.84	0.85	0.83	0.85
w6	0.83	0.85	0.85	0.86	0.86	0.86	0.86	0.85	0.84	0.86
w7	0.84	0.85	0.85	0.85	0.84	0.85	0.85	0.85	0.84	0.85
w8	0.85	0.86	0.86	0.85	0.86	0.86	0.86	0.85	0.85	0.85
w9	0.86	0.85	0.85	0.85	0.86	0.85	0.85	0.86	0.85	0.86
w10	**0.88**	0.87	0.85	0.85	0.85	0.86	0.85	0.85	0.85	0.86

Table 4: Test results obtained with ResNet 50 using data augmentation in all 10 runs, considering 2 classes every 50 epochs

Work Plan

In this chapter, the work timeline followed in the second semester is presented and compared with the initial work plan. The identified risks are then analyzed, and the planned measures to mitigate the identified risks are explained.

Work performed in the second semester

Figure 3 presents a *Gantt* chart, referent to the temporal planning of the work developed in the second semester.

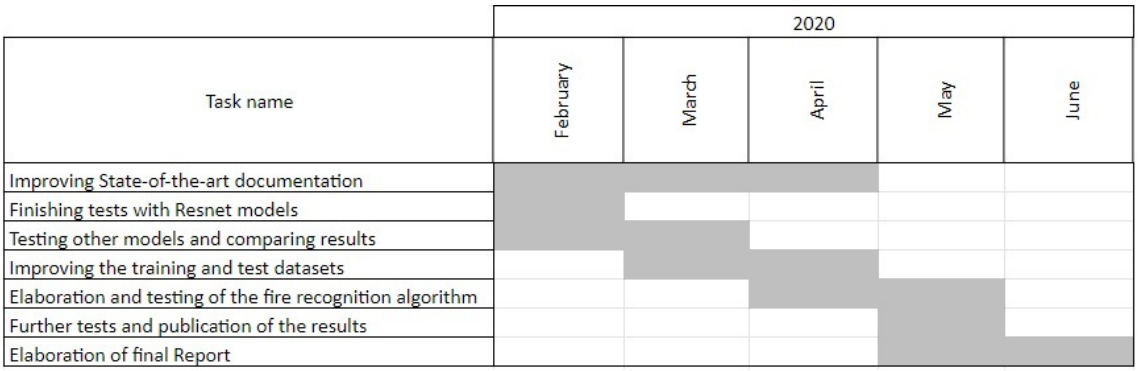


Figure 3: Second semester planned *Gantt* chart

Taking into consideration the work developed in the first semester and the results obtained, the first planned tasks aimed at continuing the study of ResNets to develop the classification approach.

One of the planned tasks was the improvement of the dataset used to achieve better results. For this purpose, the images used for training and testing were manually divided and classified, thus enabling the use of transfer learning techniques, adapting the images to the network architecture without causing distortions. Data augmentation techniques were also used, which implied a revision and more detail in the previously carried out study of the state-of-the-art. This is one of the differences between the initial planning and the actual work carried out during the second semester, which can be seen in figure 4.

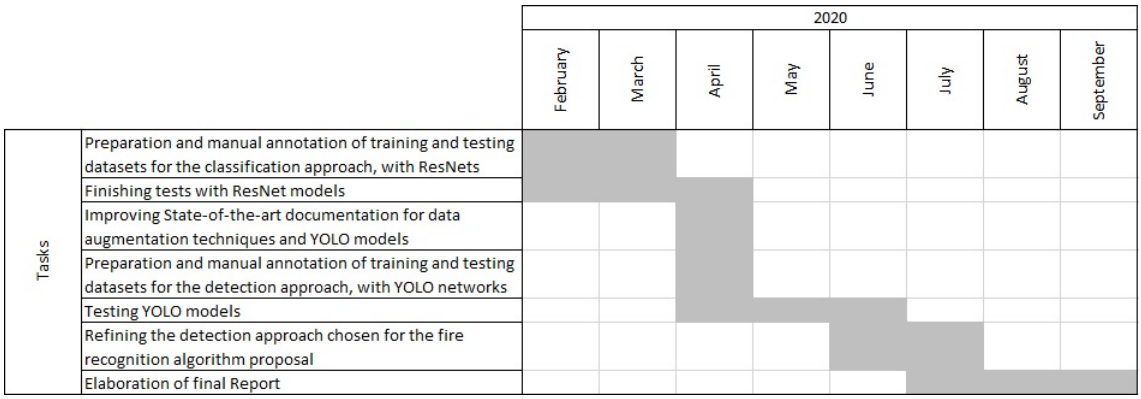


Figure 4: Second semester *Gantt* chart

In order to improve the performance of the Residual Network (ResNet) models, to avoid distortions in the images resulting from the pre-processing, the training and test images were created and annotated manually to create the used datasets. Tests were also carried out in order to optimize the training parameters of the models, and data augmentation techniques were tested, which implied a review and deepening of the study previously carried out regarding these techniques. However, it was found that the approach of classifying images as a whole might not be the most suitable for solving the problem of forest fire recognition.

The image object detection approach for fire and smoke detection in images was studied as an alternative approach. The use of new architectures, which are based on different object detection paradigms, implied the deepening of the state-of-the-art study carried out. Based on the study carried out, the You Only Look Once (YOLO) networks were studied with a process similar to ResNets: preparation and manual annotation of training data and tests followed by the optimization of the models' training parameters.

The fact that there were no annotated datasets for the YOLO networks' training, specific to the forest fire and flame detection problem, led to the use of the *LabelImg* tool and the manual annotation of each column of smoke and fire outbreak present in the images.

The process of manually annotating custom datasets, although in the case of YOLO networks was facilitated by using the annotation tool mentioned above, it is time-consuming, since the occurrence of errors can prevent the achievement of good test performance of the models. This time-consuming process led to a further divergence between the temporal planning of tasks and the work carried out.

For elaborating the proposed approach for fire and smoke recognition in images, a post-processing step of the results obtained with the YOLO networks was also proposed to obtain an image classification. Based on the results, the test parameters were adjusted in order to be able to use the intelligent fire detection system proposed in a forest environment, with two operating points to consider. Finally, the work developed was documented in the final report.

The main differences between the work planned and the work carried out in the second semester are related to the addition of tasks performed related to the detection approach. The necessary additional state-of-the-art study for image object detection and the fact that the models used are even more complex (which implies more time-consuming training) had direct implications for the time it took for each of the tasks performed. As such, it was not possible to publish the results obtained, and since the time it takes for each detection is appropriate for use in real-time applications, tests on the temporal performance of the

models were not carried out.

Risks

This section presents the identified risks and their implications for the work developed. The analysis of each risk takes into consideration its probability of occurrence and level of impact on the work developed, according to table 5. The mitigation actions proposed for each risk are also presented, as well as the strategies followed to implement them.

	Probability	**Impact**
1	Low	Marginal
2	Medium	Critical
3	High	Catastrophic

Table 5: Scale used for Risk evaluation

Risk ID	**Risk - Consequences**	**Probability**	**Impact**
R1	Training deep learning models can take a long time and might lead to delays in obtaining results	3	2
R2	Not enough available annotated data for training	2	2

Table 6: Risk identification and analysis

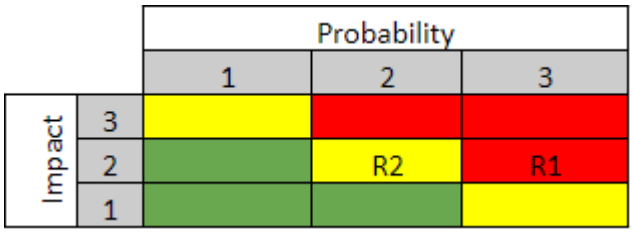


Figure 5: Summary of the risk analysis

Table 5 is a risk matrix, providing visual analysis of the impact of each risk, showing Risk 1 as the most severe.

For the risks identified, the mitigation strategies proposed were:

- Risk 1 - The use of techniques to make training more efficient, using tools like Google Colab. The training and testing of the models for both approaches were carried out using Google Colab, which allowed the use of parallel computing platforms such as CUDA and, consequently, faster training and testing.
- Risk 2 - Construction of an annotated dataset, using images from different sources, suitable for the problem.

  Following the strategy proposed, four problem-specific datasets were created and then used in the development of model classification and detection approaches (two for testing and two for testing the models).