TÉCNICO LISBOA

Automatic detection of forest fires: a deep learning approach

André de Assunção Marrucho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pedro Filipe Zeferino Aidos Tomás
Prof. Nuno Filipe Valentim Roma

Examination Committee

Chairperson: Prof. Pedro Tiago Gonçalves Monteiro
Supervisor: Prof. Pedro Filipe Zeferino Aidos Tomás
Member of the Committee: Prof. Nuno Cruz Garcia

October 2021

Acknowledgments

I would like to thank my supervisors, Prof. Pedro Tomás and Prof. Nuno Roma, and also Prof. Helena Aidos for their constant support and availability throughout the development of this thesis. They were very supportive guiding and giving constant feedback on how I could improve my work.

I would also like to thank all my friends with whom I had amazing experiences and who helped me get through this long 5-year journey. All these fantastic moments will remain in my memory.

Last but not least, I would like to thank my family for always being very supportive throughout all these years. To my girlfriend, Liliana Gomes, that has been on my side during this journey and gave me the strength to finish it. I would also like to express my gratitude to my mother, Ana Cristina Assunção, father, Fernando Marrucho, sister, Cláudia Marrucho and grandmother, Maria Aldina Assunção, for their continuous support and encouragement.

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects UIDB/50021/2020 and PTDC/EEI-HAC/30485/2017 (HAnDLE) and PCIF/MPG/0051/2018 (ResNetDetect). I would also like to thank Instituto de Engenharia de Sistemas e Computadores Inovação (INOV) for providing the labelled smoke dataset used in this work.

Abstract

Forest fires have a devastating impact in several countries, causing serious environmental, social, and economic damage. The effect of global warming is turning extreme natural disasters into normal events and forest fires are no exception. Several strategies have been developed to detect early-stage fires in recent years, namely the use of surveillance cameras in forest areas to autonomously monitor and detect any incident. However, these autonomous systems often signal a high number of false positives, demanding several manual checks on the images. The recent use of deep neural networks to perform image classification in a wide variety of domains has proven to be efficient and accurate. This dissertation proposes a method to detect early-stage fires, using pre-trained state-of-the-art deep learning architectures, together with an image generation approach that may be used to overcome the problem of limited data in smoke/fire datasets. Tested with real smoke and fire images, this approach can be applied in several real-world scenarios such as the capture of images by surveillance cameras or drones to detect emerging fires. The experimental results in image generation show the ability to generate small smoke plumes that can be merged with normal images to create artificial smoke images. Regarding image classification, the model is able to detect early-stage fires with 98.2% accuracy, using images from a variety of locations.

Keywords

Smoke detection, early fire detection, deep learning, forest fires, image generation

Resumo

Os fogos florestais têm um impacto devastante em vários países, causando sérios danos a nível ambiental, social e económico. O efeito do aquecimento global tem transformado desastres ambientais raros em eventos normais e os fogos florestais não são uma exceção. Várias estratégias têm sido desenvolvida para detetar fogos nas suas fases iniciais através, por exemplo, do uso de câmaras de vigilância em zonas florestais. Contudo, estes sistemas autónomos sinalizam, frequentemente, um elevado número de falsos positivos, exigindo muitas vezes uma verificação manual das imagens. O uso recente de redes neuronais para classificar imagens numa ampla variedade de domínios tem provado ser eficiente e preciso. Esta dissertação propõe um método para detetar fogos nas suas fases iniciais, através do uso de redes neuronais pré-treinadas, assim como uma estratégia para gerar imagens de forma artifical e que pode ser usada para ultrapassar o problema da quantidade de dados limitada existente em alguns conjunto de dados. Os modelos foram testados com várias imagens reais de fogos e podem ser aplicados em vários cenários do mundo real, como a captura de imagens através de câmaras de vigilância ou drones para detetar pequenos fogos. Os resultados experimentais na área da geração de imagens mostram a capacidade em gerar pequenas colunas de fumo que podem ser inseridas noutras imagens para criar imagens artificiais com fumo. Relativamente à classificação de imagens, o modelo é capaz de detetar com sucesso fogos que estão nas suas fases iniciais, com 98.2% de precisão, usando imagens recolhidas em várias localizações.

Palavras Chave

Deteção de fumo, deteção de fogos emergentes, aprendizagem profunda, fogos florestais, síntese de imagens

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.2 CICLOPE Project . . . 3

1.3 Objectives . . . 4

1.4 Contributions . . . 5

1.5 Thesis Outline . . . 7

2 Background & Related Work **9**

2.1 Basic Concepts . . . 10

2.2 Deep Learning Classification Architectures . . . 11

2.3 Image Generation Architectures . . . 14

2.4 Fire Detection . . . 16

2.5 Data Augmentation and Transfer Learning . . . 19

2.5.1 Image-to-image translation . . . 19

2.5.2 Few/One-shot Learning . . . 20

2.5.3 Transfer Learning . . . 22

2.6 Summary . . . 23

3 Deep Neural Models for Image Classification and Generation **25**

3.1 Image Pre-Processing . . . 27

3.1.1 Image division . . . 28

3.1.1.A Normal grids . . . 28

3.1.1.B Overlapped grids . . . 29

3.2 Deep Learning Classification Architectures . . . 29

3.2.1 Proposed framework . . . 30

3.2.2 Xception . . . 31

3.2.3 InceptionResNetV2 . . . 33

3.2.4 EfficientNet . . . 33

3.3 StyleGAN . . . 34

3.4 Summary . . . 37

4 Dataset Description **39**

4.1 INOV Dataset Description . . . 40

4.2 Image Classification Datasets . . . 42

4.2.1 Train Dataset . . . 42

4.2.2 Test Dataset . . . 42

4.3 Image Generation Datasets . . . 43

4.4 Summary . . . 43

5 Experimental Evaluation **45**

5.1 Image Generation . . . 46

5.1.1 Metric . . . 46

5.1.2 StyleGAN2-ADA . . . 47

5.1.3 Smoke Images . . . 48

5.1.4 Smoke Plumes Images . . . 49

5.2 Image Classification . . . 51

5.2.1 Metrics . . . 51

5.2.2 Experiments . . . 52

5.2.3 Best Results Analysis . . . 54

5.3 Discussion . . . 56

6 Conclusion **59**

6.1 Contributions . . . 60

6.2 Future Work . . . 60

Bibliography **61**

List of Figures

1.1 Area burnt by forest fires in European countries . . . 2
1.2 Scheme of the image acquisition and prediction system . . . 3
1.3 Examples of small, emerging fires . . . 4
1.4 Example of a small smoke column in the horizon . . . 5
1.5 Comparison between the dataset used in this and other works . . . 6

2.1 VGG16 architecture . . . 11
2.2 InceptionV1 module . . . 12
2.3 InceptionV2 module . . . 13
2.4 ViT architecture . . . 14
2.5 Example of the YOLOv4 object detection . . . 15
2.6 Sample images generated by the BigGAN network . . . 16
2.7 Example of an input fire image . . . 18
2.8 Testing results with the corresponding bounding boxes . . . 19
2.9 Example of the CycleGAN application using a season transfer effect . . . 21
2.10 Matching Networks architecture . . . 22

3.1 Overview of the proposed system . . . 26
3.2 Example of a smoke image with the fire location . . . 27
3.3 Example of a 5x4 grid image with the fire location . . . 28
3.4 Example of a 5x4 overlapped grid image with the fire location . . . 29
3.5 Deep learning architecture overview using the first transfer learning method . . . 30
3.6 Deep learning architecture overview using the second transfer learning method . . . 31
3.7 Xception architecture . . . 32
3.8 InceptionResNetV2 architecture . . . 33
3.9 EfficientNet network compared to other architectures . . . 34
3.10 StyleGAN generator architecture . . . 35

3.11 Examples of images produced by the StyleGAN generator with the FFHQ dataset 36
3.12 Examples of smoke plumes extracted from fire images 37

4.1 Examples of non-smoke images in the dataset 40
4.2 Examples of smoke images in the dataset 41
4.3 Examples of smoke hotspots in the DSH dataset 44

5.1 Examples of generated images using the original StyleGAN2-ADA architecture 48
5.2 Examples of 256x256 smoke generated images 50
5.3 Examples of 64x64 smoke plumes generated images 51
5.4 Training and validation accuracy. 55
5.5 Examples of False Positives 55
5.6 Examples of True Positives with fire location 56
5.7 Examples of True Negatives 57

List of Tables

4.1 Distribution of smoke and non-Smoke images by tower 41
4.2 Distribution of smoke images in the test dataset by smoke area 42
4.3 Summary of the training datasets 43

5.1 Comparison of the FID metric between the StyleGAN2 and StyleGAN2-ADA architectures 47
5.2 Results using StyleGAN2 using the full smoke DS dataset for training 48
5.3 Results using StyleGAN2 using the smoke plumes DSH dataset for training 49
5.4 Evaluation results for the TN-1 method. 53
5.5 Evaluation results for the TN-2 method. 53
5.6 Total number of TP, TN, FP and FN 54
5.7 Results comparison with other related works 57

Acronyms

AdaIN	Adaptive Instance Normalization
EU	European Union
CCTV	Closed-circuit Television
CMC	Control and Management Center
CNN	Convolutional Neural Network
FFHQ	Flickr-Faces-HQ
FID	Fréchet Inception Distance
INOV	Instituto de Engenharia de Sistemas e Computadores Inovação
kNN	k-Nearest Neighbors
FN	False Negatives
FP	False Positives
GAN	Generative Adversarial Network
TN	True Negatives
TP	True Positives
UAV	Unmanned Aerial Vehicle
VAE	Variational AutoEncoder
vCPU	Virtual Centralized Processing Unit

1

Introduction

Contents

1.1 Motivation 2
1.2 CICLOPE Project 3
1.3 Objectives 4
1.4 Contributions 5
1.5 Thesis Outline 7

1.1 Motivation

Wildfires are one of the greatest hazards in forests. Forest fires threaten not only the forest's fauna and flora but also the biodiversity of the entire region. Beyond the environmental and economic impact, uncontrolled fires often pose a danger to local populations, when they reach a village's vicinity. The effect of global warming has caused, in the past years, a global average temperature rise, contributing to an increase in the number and dimension of wildfires.

In Portugal, forest fires are a very common phenomenon in the summer season. They are one of the most severe natural disasters in our country not only because of the high frequency and dimension they reach but also because of the devastating effects they cause. According to Pordata [1], Portugal has the highest burnt area of the European Union (EU) in three of the last four years with available data (2016-2019). In 2017 alone, Portugal had 541 thousand acres of burnt area, which represents three times more than Spain, the second most affected country. Despite being one of the smallest countries, in area, affected by wildfires in the EU, Portugal is still devastated, every year, by thousands of forest fires. And, although a high effort has been put recently in prevention measures, creating new laws, and making people more aware of the dangers that forest fires pose to everyone, we are still far from controlling the consequences of wildfires. Figure 1.1 displays the data regarding the area burnt by forest fires in the most affected European countries, between 2016 and 2019.

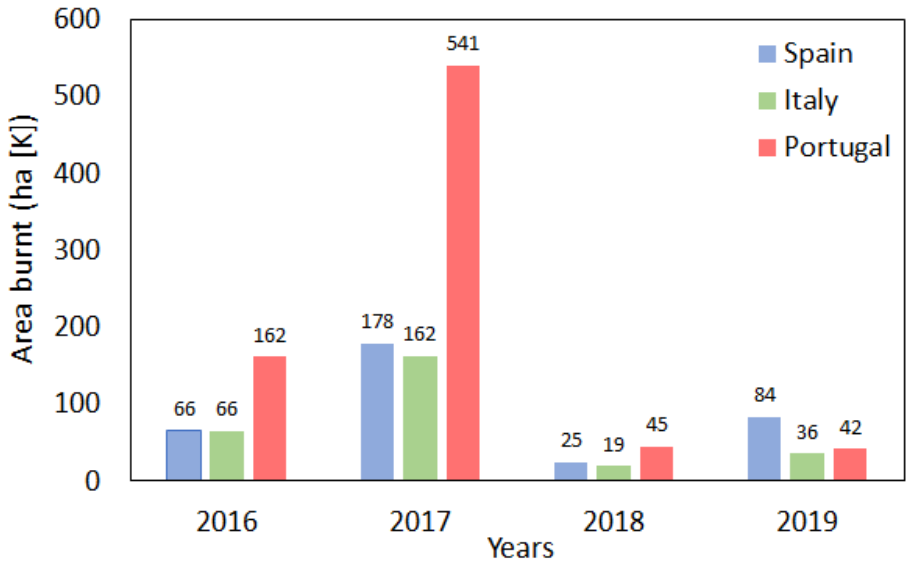

Figure 1.1: Area burnt by forest fires in the most affected European countries between the 2016 and 2019 period. Hectares [K] represents the forest area burnt, divided by 1000. Data extracted from [1].

Regarding Europe and according to the European Environment Agency [2], more countries have suffered large forest fires in 2018 than ever recorded before, including in northern and central Europe, regions not typically affected by wildfires. "Many of the recent extreme fire episodes and devastating

fire seasons in Europe were driven by severe weather conditions, with record droughts and heat waves occurring in the spring and summer of 2017 and 2018 for instance." To minimize the impact of wildfires, it is very important to develop new ways to detect them at an early stage so that they can be more easily controlled and extinguished.

1.2 CICLOPE Project

The CICLOPE project [3] is a wildfire monitoring system developed by researchers at Instituto de Engenharia de Sistemas e Computadores Inovação (INOV), allowing "automatic detection of emerging wildfires and instantaneous first response trigger". The CICLOPE system covers, at the moment, more than 1 million acres of land in Portugal. The use of individual and simultaneous images captured in the visible and infrared wavelength allows to capture images in day and night and in nearly all weather conditions.

The CICLOPE system was designed to work in every location, having independent power supplies and supporting several means of communication. The system is composed of several surveillance and data acquisition towers, where the information is acquired by the surveillance cameras. These cameras can cover a wide area of land, with their high zoom range and ability to rotate 360º. The images captured are then sent to the Control and Management Center (CMC) to be monitored. The application in the CMC can be used for the automatic detection of fires, which allows identifying smoke columns and flames even in the most adverse meteorological conditions. The use of this mechanism reduces manpower requirements and can be effortlessly applied to a large collection of surveillance cameras, allowing to cover a wide range of land.

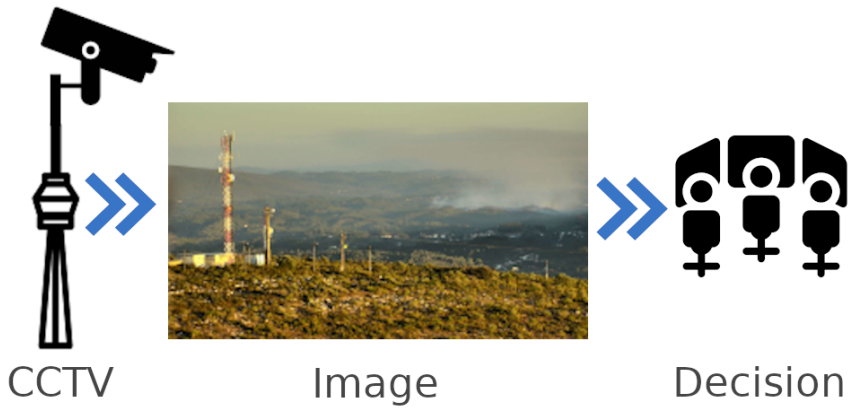

Figure 1.2: Overall scheme of the image acquisition and prediction system. The images are captured by surveillance cameras located in forest areas. The focus of this work is the prediction process that assigns a "smoke" or "non-smoke" label to the image. Later, if the output label is "smoke", the image should be manually checked.

However, despite the advantages of having an automatic detection system, these systems some-

times do not reach human-level accuracy when detecting early-stage fires, mainly because of the high number of false positives. This fact reduces the usefulness of the automatic detection mechanism, requiring more manpower to operate the CICLOPE system and, potentially, missing some important early-stage smoke columns.

The images used to perform training and testing in this work were taken from the cameras that are set up on the top of the surveillance and data acquisition towers located in forest areas. This work focuses in developing an automatic detection system capable of classifying these images as smoke or non-smoke. A global overview of the system is illustrated in Figure 1.2.

1.3 Objectives

The main goal of this project is to develop an image classification mechanism that can be applied to the images that are collected from the surveillance cameras. These mechanisms should have good accuracy when detecting not only large size fires but especially small, emerging smoke columns. The early detection of fires is a very important aspect to control and prevent them to become larger and more dangerous phenomenons. The detection of small fires can also be a very difficult task as, sometimes, just a very tiny smoke column is shown on the horizon, that even humans struggle to spot, as exemplified in the next set of examples.

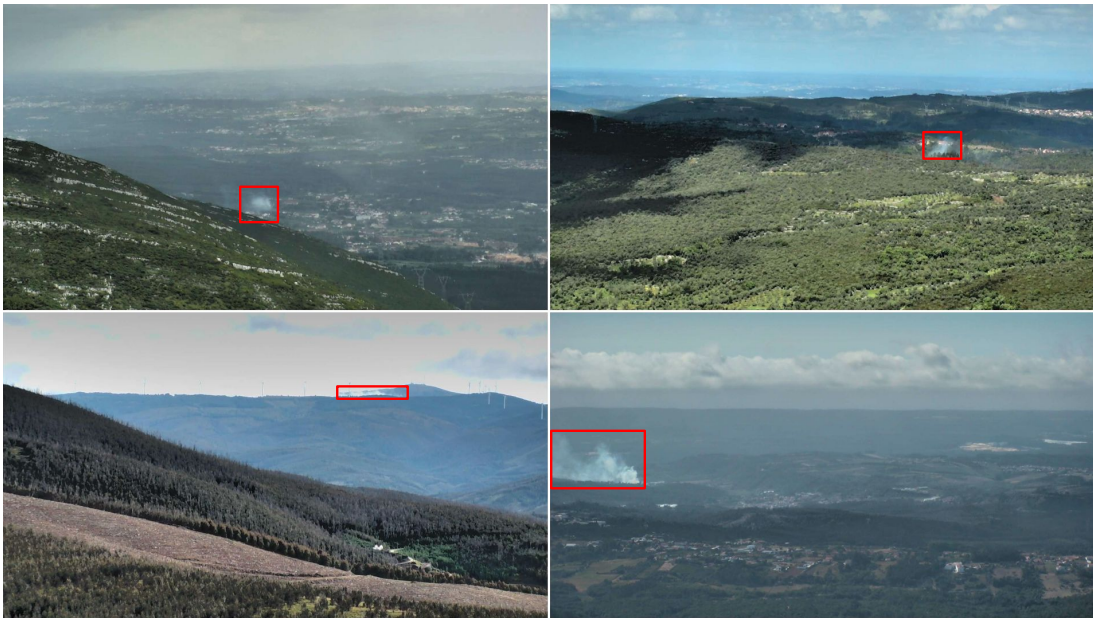

Figure 1.3: The first three images include small, emerging fires. In the last image, it is possible to observe a medium-sized fire.

To develop this project, INOV provided a dataset with images taken from their Closed-circuit Television (CCTV), with both fire and non-fire images. A few examples of images that were taken from this

dataset will now be discussed. The first three images in Figure 1.3 are typical examples of emerging fires, with the presence of small white plumes that indicate the presence of a fire in its first stages. The identification of the first stages of a fire is the main goal of this project. In the last image, a medium-sized smoke column also indicates the presence of a fire. Medium and large-sized fires are usually the easiest for the system to detect but they are also the most useless because, most likely, someone has already identified them and contacted the authorities.

Taking a look at the third example, Figure 1.4, anything unusual can be immediately spotted. However, when looking closely at the top right corner of the image, a tiny smoke area, that will later become a larger fire, can be identified. The main goal of this thesis is not to identify large flames but rather these smoke columns that indicate the presence of fires in images that cover a wide range of forest areas. The most challenging obstacle will be the similarity between these small smoke plumes and clouds or fog.

Figure 1.4: A very small smoke column in the top right corner can be seen.

1.4 Contributions

Several methods have been used to perform fire detection, namely using computer vision methods. A common technique is to consider consequent frame sequences and locate regions of movement that could indicate that a fire is constantly changing its size [4]. A few other computation vision methods [5, 6] analyse specific fire features, such as colour, area size and spatial distribution and combine them to produce a global output prediction.

The main difference between this and other works is the concern in detecting early-stage fires. This work focuses on detecting small smoke columns rather than large areas of smoke or flames. To illustrate

this difference, Figure 1.5 includes a sample image that was used for smoke detection in this work compared to smoke/fire images used in other papers.

Figure 1.5: The first image is a sample from the dataset used in this work. The other three images were taken from other papers ([7], [8] and [9] respectively) related to smoke/fire detection.

Hence, the main contributions of this MSc thesis are summarized as follows:

- Development of a deep learning approach to detect early-stage fires in RGB images. To perform the image classification stage, the following deep neural networks were used: Xception [10], InceptionResNetV2 [11] and EfficientNet [12] architectures pre-trained in the ImageNet [13] dataset. The proposed approach includes a fine-tuning phase using a custom dataset with target domain images, i.e., smoke images. During the training phase, two different approaches were explored: the detection of smoke in the entire image; and the division of the image in blocks and, thus, detecting smoke in these individual blocks.

- One of the main contributions of this thesis is to explore new data augmentation techniques to generate new smoke/fire images. These generated images can be used not only to improve datasets with limited data but also to fill some gaps that may exist in large datasets, such as the lack of night fire images, fog images, etc. The image generation stage was performed training the StyleGAN2-ADA architecture [14] with smoke images. Due to time restraints, these images were not included in the training datasets when performing the image classification stage.

The potential applications of this thesis in real-world scenarios include early smoke and fire detection from images that can be captured from various sources, such as surveillance cameras or Unmanned Aerial Vehicles (UAVs).

1.5 Thesis Outline

This thesis is organized as follows: Chapter 2 presents the state-of-the-art techniques in deep learning architectures for image classification and generation, evaluation metrics and data augmentation. It will also feature a few methods used to detect fire in images. Chapter 3 details the pre-processing work performed in the dataset images and also the architectures that were used for both image classification and generation purposes. Chapter 4 describes the dataset that was provided by INOV as well as several others that were created to perform the experiments. Chapter 5 covers the experimental methodology and results of the adopted approach. Finally, Chapter 6 presents the main conclusions of this work and proposes insights that can be explored for further work and improvement.

2

Background & Related Work

Contents

2.1 Basic Concepts 10
2.2 Deep Learning Classification Architectures 11
2.3 Image Generation Architectures 14
2.4 Fire Detection 16
2.5 Data Augmentation and Transfer Learning 19
2.6 Summary 23

This chapter presents some of the relevant state-of-the-art work in the deep learning field, transfer learning tasks and also other techniques that have been used for fire detection. In particular, Section 2.1 introduces some basic concepts in the artificial intelligence field, giving a brief insight on machine learning and deep learning, mentioning some of its applications in our daily life. Section 2.2 overviews the state-of-the-art in deep learning models for image classification, with particular emphasis on models that have shown the best results in popular datasets, such as ImageNet, CIFAR, etc. Section 2.3 presents a few architectures that have been used to generate fake images and that can be used for data augmentation. Section 2.4 presents some of the techniques that have been designed to detect the presence of fires. Section 2.5 describes a couple of approaches that have been developed to deal with limited training data. These approaches include the use of data augmentation and transfer learning. Finally, section 2.6 provides a summary of this chapter.

2.1 Basic Concepts

From self-driving cars to speech recognition, machine learning has had a strong impact, particularly in recent times, on several aspects of our society. Machine learning is defined as being a set of algorithms that can learn and improve from experience, without the need to be explicitly programmed. The main goal is to allow computers to learn from a set of data and, without any kind of human intervention, adjust their actions accordingly.

In an attempt to overcome the limitations of traditional methods used in machine learning (e.g., decision trees, support vector machines, etc), deep learning has emerged. Deep learning is a sub-field of machine learning inspired by the behaviour and structure of biological neural networks, often referred to as artificial neural networks. Deep learning algorithms attempt to make human-level decisions by using multi-layered neural networks and vast amounts of data, which sometimes demand high computational power. The use of deep learning algorithms has shown great results in several fields, namely computer vision, speech recognition, natural language processing, and many others. On the other hand, in the past few decades, the advances in hardware and software components have allowed the development of more complex deep learning algorithms that can handle complicated domains. In 2016, AlphaGo, developed by DeepMind, was the first computer program to defeat a Go world champion without handicap.

When it comes to image classification tasks, Convolutional Neural Networks (CNNs) have proven to be very effective and produce good results. The input of a CNN is an image with a certain height, width, and dimension. Each input image passes through a series of convolutional layers (that try to extract relevant features), pooling layers (to reduce the dimension of the feature maps), fully-connected layers (to combine the features), and finally a softmax layer that will output the probability distribution over every class of the domain.

2.2 Deep Learning Classification Architectures

Image classification is a task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label based on a probability distribution. Typically, image classification refers to the case in which only one object appears in the image. On the other hand, the goal of object detection involves both classification and localization tasks and it is typically used when multiple objects are present in an image.

When it comes to image classification, several deep learning architectures have been introduced in the past few years. For example, the ResNet architecture [15] introduces the concept of residual networks, where shortcuts are used to jump over some layers, allowing to create deeper networks. Similarly, the DenseNet [16] network is an extension of ResNet that uses dense blocks, where each layer takes all preceding feature-maps as input. This section will detail a few deep learning architectures that have been developed to perform image classification.

VGGNet is a Convolutional Neural Network that was proposed by Simonyan *et al.* 2015 [17]. During training, the architecture takes as input a fixed-size 224×224 RGB image. The only image pre-processing step performed is subtracting the mean RGB value from each pixel. After that, the training images are passed through a stack of convolutional layers with small 3×3 filters followed by three fully connected layers. The first two fully connected layers have 4096 channels each, and the third has 1000 channels (one for each class). Finally, there is a softmax layer for classification purposes. All hidden layers have ReLU as their activation function. The most popular variations of VGGNet are VGG16 (Figure 2.1) and VGG19. The VGG16 architecture has a total of 13 convolutional layers, whereas the VGG19 has 16 convolutional layers.

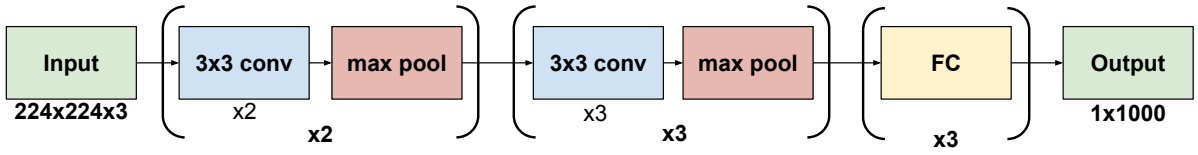

Figure 2.1: VGG-16 architecture. The number of times (N) that each stack of layers is repeated, is indicated by **xN**.

The most straightforward way of improving the performance of deep neural networks is by increasing their size. This includes both increasing the depth (the number of layers) of the network and its width (number of units at each level). However, this solution has two main drawbacks. Bigger usually means a larger number of parameters, which makes the network prone to overfitting and also more difficult to train. Another drawback is the increased use of computational resources. To overcome this problem, a first version of the Inception architecture was developed by Szegedy *et al.* 2015 [18]. In this architecture, each level has filters with multiple sizes to capture different features in the input. Basically, in each module, a convolution is performed on the input, with 3 different filters (1×1, 3×3, and 5×5). The

outputs are concatenated and sent to the next inception module. This makes the network wider rather than deeper. However, 3×3 and 5×5 convolutions can be expensive on top of convolutional layers with a large number of filters. To overcome this problem, an extra 1×1 convolution is added to reduce the dimensionality of the feature maps before the 3×3 and 5×5 convolutions. In general, an Inception network is a network consisting of Inception modules, shown in Figure 2.2, stacked upon each other.

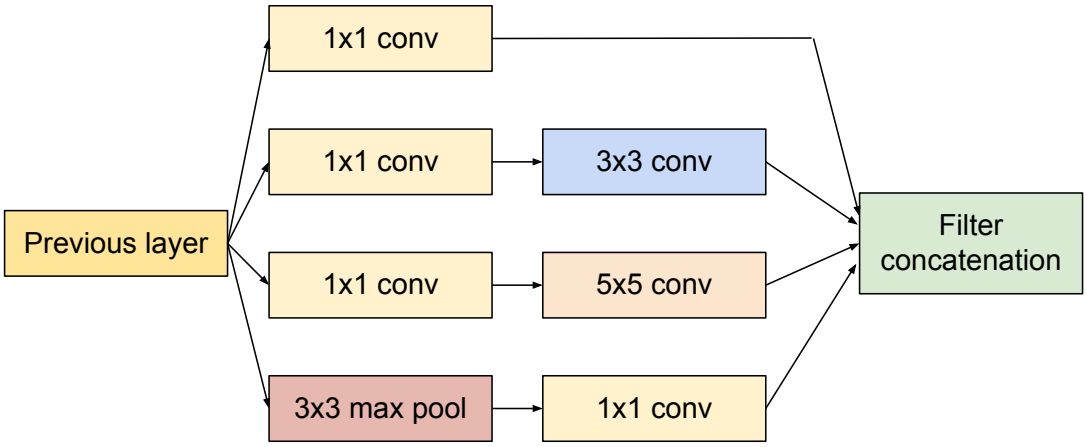

Figure 2.2: Inception module with dimension reductions. Prior to the 3×3 and 5×5 convolutions, a 1×1 convolution is performed in order to reduce the input dimension.

Both Inception-v2 and Inception-v3 were introduced by Szegedy *et al.* 2016 [19] to mitigate some of the problems of the previous version. Firstly, it attempts to avoid representational bottlenecks, since extreme compression of the input can lead to the loss of important features. Secondly, it improves the computational complexity by using factorization techniques. The 3 main architectures changes that were made in the Inception-v2 network were the following:

- Replace the 5×5 convolution with two layers of 3×3 convolution. This reduces the computational cost, as the number of parameters is reduced by 28%.
- Factorize the $n \times n$ convolution with two layers of $1 \times n$ and $n \times 1$ convolutions. For example, using a 1×3 convolution followed by a 3×1 convolution is equivalent to using a 3×3 convolution. However, the two-layer solution is 33% cheaper.
- Filter banks were expanded, to make the model wider rather than deeper. This avoids representational bottlenecks, which translates into a lower loss of information (Figure 2.3).

The Inception-v3 model shared the same base architecture as the previous one, with a few changes which include the use of the RMSprop optimizer, factorization of 7×7 convolutions, the use of batch normalization in the auxiliary classifier and label smoothing.

Dosovitskiy *et al.* 2020 [20] propose the implementation of transformers (used in NLP tasks) for image classification. First, the original image *x* is reshaped into a sequence of flattened 2D patches,

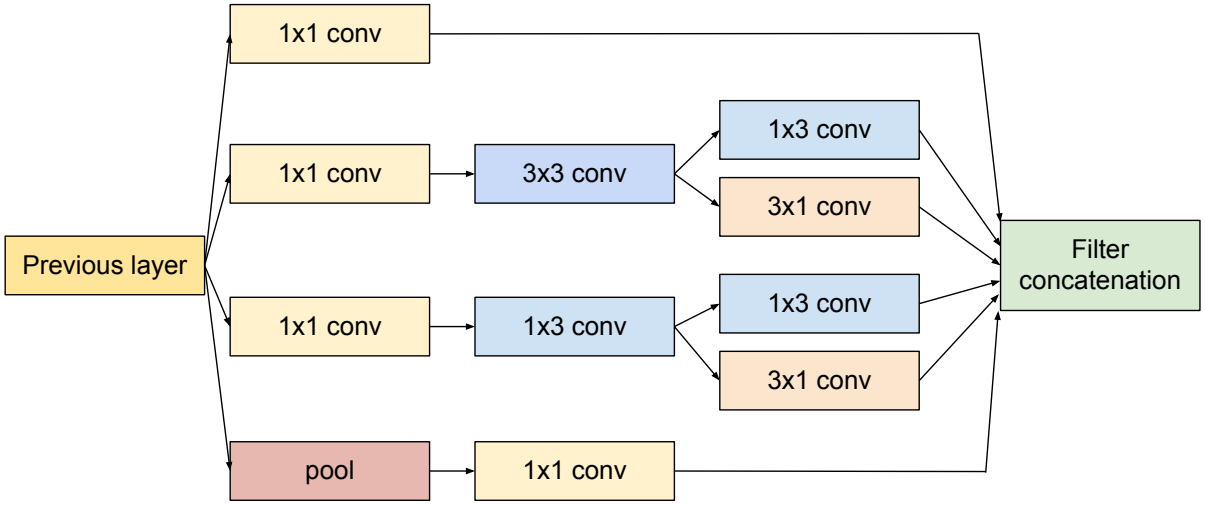

Figure 2.3: Inception-v2 module.

where (P, P) is the resolution of each image patch. A linear projection is then used to map each patch to the model dimension D, referred to by the authors as patch embeddings. A learnable embedding, containing information about the class of the image, is prepended to the patch embeddings. Finally, to retain positional information about all patches, a position embedding is added. These embedded patches are then passed to the encoder, and finally, the classification token is used to predict the class of the image, as shown in Figure 2.4. This ViT architecture is pre-trained on large datasets and then fine-tuned to smaller tasks. With several experiments, the authors discovered that the larger the pre-trained dataset is, the better results the model yield. The best results were achieved using JFT300M dataset for pre-training. They achieved slightly better accuracy results in several known datasets (ImageNet, CIFAR-100, Oxford Flowers-102) than state-of-the-art models (ResNet and EfficientNet-L2 were used for comparison), while also needing significantly less computational power to pre-train the ViT model.

Despite being designed for object detection rather than image classification, the YOLO architecture has been the standard when it comes to locating and classifying objects in videos. The first version of the YOLO architecture was proposed in Redmon *et al.* 2016 [21] and it was designed for real-time object detection. The architecture resizes the input image to a 448×448 dimension array and divides it into an $S \times S$ grid (S=7). If a certain grid has the centre of an object in it, that grid cell is responsible for detecting that object. Each grid cell predicts B (B=2) bounding boxes and the respective confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object. Each bounding box consists of 5 predictions: x, y, w, h, and confidence: (x, y) represents the coordinates of the centre of the object; (w, h) represent the width and weight of the bounding box; and confidence represents the Intersect Over Union (IOU) between the predicted box and any ground truth box. The YOLO architecture is fast compared to the existent real-time object detection models and it is

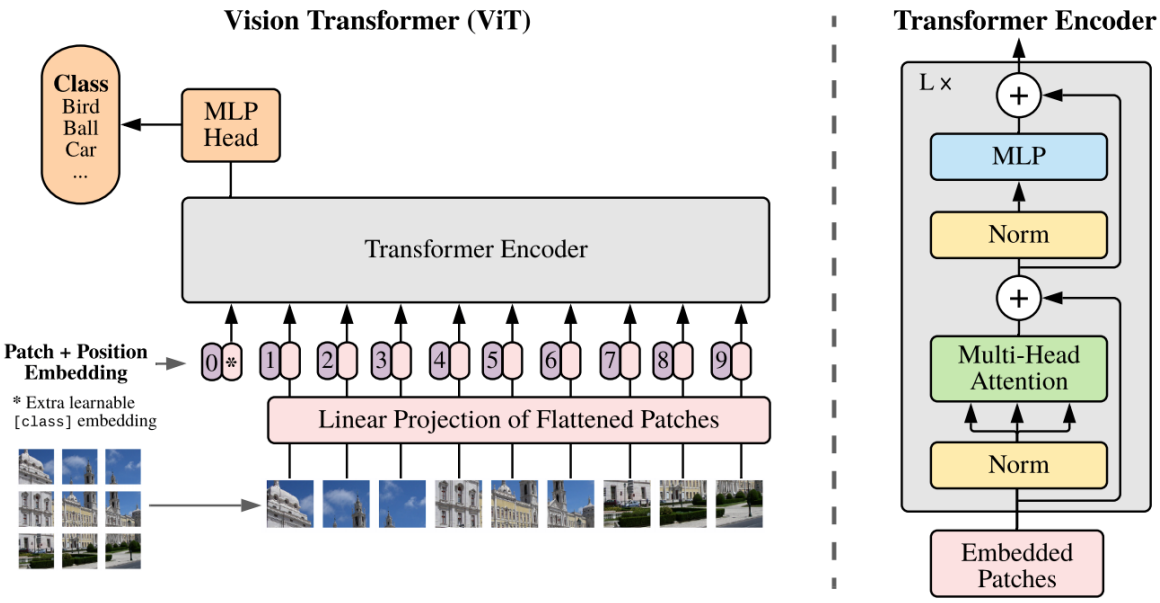

Figure 2.4: Architecture overview, (Extracted from [20]).

also able to generalize well enough to be applied to new domains while achieving good results.

The most recent version of YOLO - YOLOv4 - was proposed in Bochkovskiy *et al.* 2020 [22] and introduces several improvements to previous versions. These improvements include the use of new data augmentation techniques like Mosaic that mixes 4 different images and produces a single output image and new regularization techniques, such as DropBlock. This latest version, YOLOv4, was able to produce state-of-the-art results in real-time object detection, averaging 43.5% average precision (AP) at 65 fps.

2.3 Image Generation Architectures

Image generation refers to the task of generating new images based on an existing dataset. Two of the most popular methods to create new images are Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs).

An autoencoder is composed of two connected networks: encoder and decoder. First, the encoder compresses the original image into a lower dimension (latent space). Then, the decoder decompresses that data and tries to reconstruct the original image. The reconstruction loss measures how distinct the reconstructed image is compared to the original one. However, in Variational AutoEncoders (Kingma *et al.* 2014 [24]), the reconstruction of the original input is not the only purpose, but also to introduce new variations to it from a continuous space. Since the latent space in normal autoencoders is not

Figure 2.5: YOLOv4 object detection, (Extracted from [23]).

often continuous, a new term is added to the loss function, the KL divergence, which helps to optimize the probability parameters (mean and variance) to resemble the target distribution. To generate new images from a certain class, the VAE is trained with images belonging to that class. Then, a vector (with the same dimension as the latent space) is initialized with random values and passed to the decoder. Using this technique, new images that can be used to augment the dataset are produced. The main disadvantage of VAE's is that the generated images are often blurry, especially when training the model with a small dataset.

An alternative approach to synthesize new images is by using Generative Adversarial Networks. GANs were introduced in Goodfellow *et al.* 2014 [25] and are composed of two parts: the generator, which learns to generate new data as close as possible to real images from the dataset; and the discriminator, which learns to distinguish real examples from the ones produced by the generator (fake examples). To train the network, the first step consists of providing a large number of examples until the discriminator achieves acceptable accuracy in classifying the images. Then, in the second training step, the generator is initialized with random input, sampled from the latent space, and tries to fool the discriminator with the image produced. Then, both the first and second training steps are repeated to keep training the discriminator and the generator. This training loop is repeated until the desired results are achieved. This type of network is usually very unstable to train but can generate realistic results.

The developments of several GANs models over the years has made possible the creation of high-quality images. The StyleGAN architecture was introduced in Karras *et al.* 2018 [26]. The model uses a Baseline Progressive GAN and starts by training the network in smaller resolutions (4×4). Once both the generator and discriminator are stable, the resolution of the input images is progressively increased until

the desired output resolution is achieved. Miyato *et al.* 2018 [27] propose a normalization technique, spectral normalization, to be used in GANs to stabilize the training of the discriminator. By using this normalization technique, the networks are able to generate more diverse examples and achieve better experimental results when compared to regular GANs. Brock *et al.* 2019 [28] introduce the BigGAN network that aims to generate higher-resolution images by scaling up GANs. The BigGAN is designed to be a class-conditional GAN, where the input of the network is not only a point from the latent space but also includes information regarding the desired image class. The BigGAN presents a technique to make architectural changes to regular GANs, by increasing the number of parameters, using larger batch sizes and other methods. By introducing a truncation trick that resamples a point in the latent space, the network is able to generate higher-quality images at the cost of variety. Figure 2.6 displays a few class-conditional samples generated using the BigGAN architecture.

Figure 2.6: Sample images generated by the BigGAN network, (Extracted from [28]).

2.4 Fire Detection

Several methods have been used to perform fire detection, namely using computer vision techniques, that aim to extract high-dimensional data from the real world in order to produce symbolic information, which in this case is the detection of smoke or fire in images. True 2009 [4] considers consecutive frame sequences to locate regions of movement and extracts the pixels corresponding to these regions using a perceptron. Then, texture analysis is performed to confirm that these moving regions have the temporal and motion characteristics of fire. Borges *et al.* 2010 [5] proposes a method to analyse frame-to-frame changes of specific low-level features describing potential fire regions. These features are colour, area size, surface coarseness, boundary roughness, and skewness. In Qi and Ebert 2009 [6], the authors introduce an algorithm that uses not only colour and movement but also analyses the temporal variation of the fire intensity and the tendency of the fire to be grouped around a central point, to detect the presence of fire in video sequences.

More recently, with the application of CNNs to perform image classification in a wide range of domains, new methods have been introduced to detect fire or smoke in images. Lin *et al.* 2019 [29] uses

a faster region-based CNN to generate suspected smoke boxes in target frames that are picked from video sequences at a fixed interval. Then, a 3D CNN is used to perform smoke recognition in the selected frames and signal the existence of smoke. In Yin *et al.* 2017 [9], the authors combine the use of image normalization with a CNN to detect the presence of smoke in images. The image pre-processing stage reduces the variance of images and enhances inherent characteristics of images, by removing the influence of illumination. Then, the CNN is responsible for performing the image classification stage. In Muhammad *et al.* 2018 [30], the authors use the AlexNet architecture [31] pre-trained in the ImageNet dataset to detect fire in images. The network is then fine-tuned using a target dataset that includes fire images.

This section will detail a few works that have shown good results when detecting fire and smoke in images using both computer vision techniques and neural networks.

Genovese *et al.* 2011 [32] propose a mechanism for real-time smoke detection using low quality images captured under visible light. The proposed approach extracts different features from the original images and then uses a classifier to assign one of the two labels, “fire” and “nofire”, to each pixel considering a fixed number of consecutive frames. The feature extraction method includes moving region detection, smoke-colour detection and growing region detection. The experimental results showed that using neural networks as classifiers proved to be more accurate and also required less computational time than k-Nearest Neighbors (kNN) classifiers. The authors tested the approach in three different datasets: low-quality frame sequences, medium-quality frame sequences and synthetic frame sequences. The tests performed in the three datasets resulted in accuracy values around 99%. When testing the first dataset with adverse conditions, such as fog, decreased luminance or noise, there was only a small increase in the classification error (0.15%).

Labati *et al.* 2013 [33] propose an improved version of the previous classification method, with the addition of a synthetic image generation algorithm. The proposed wildfire smoke simulation method aims to generate smoke plumes to be merged with normal images and create artificial smoke images, which can later be used to train the classifiers. The first step consists in computing an artificial smoke plume. Then, in order to simulate the presence of wind, an external force is applied to the smoke plume. Finally, the smoke plume is merged with a normal image frame to produce the artificial image. With the experimental evaluation, the authors concluded that the use of simulated frame sequences can increase the accuracy and generalization capability of the wildfire smoke detection approach. The datasets used in this and the previous work are not public but the examples included in the papers show images with small smoke plumes, very similar to the ones used in this thesis. Despite this fact, the number of smoke examples represents only about 2% of the total number of images in each of the three datasets used for testing purposes.

Frizzi *et al.* 2016 [8] propose the use of CNN for identifying fire in videos. The CNN operates

directly on a raw RGB image to learn a set of visual features. The CNN architecture combines several convolutional layers, to extract the features, with pooling layers, to reduce the input dimension. The last layer is a fully connected layer that outputs a probability distribution over the three possible outputs: negative, fire and smoke. The dataset used for testing is composed of 1427 fire images, 1758 smoke images and 2399 negative images. The few examples of images included in the paper are large fires, with flames and large areas of smoke (see Figure 2.7). Thus, the main goal of the authors is the detection of fires of any dimension in images, rather than the detection of emerging fires such as in the case of this dissertation. The final classification accuracy for the test dataset was 97.9%.

Figure 2.7: Example of an input fire image, (Extracted from [8]).

In Jiao *et al.* 2019 [7], the authors propose the use of UAVs to monitor and detect forest fires in real-time. The UAV is equipped with a camera to capture the images and an onboard computer that can perform real-time image processing. If the UAV identifies a potential fire spot, it relies on that data to a ground station that would detect and diagnose a forest fire. The onboard computer uses a modified version of the YOLOv3 algorithm [34] in order to perform the fire detection. During the experiments, the UAV was capable of reaching 3.2 frames per second when performing the image classification in a 1920×1080 video sample. When the authors tested the model with 60 images, the detection rate reached 83%. Once again, the examples shown by the authors represent large size fires, with either big flames or large smoke columns, as exemplified in Figure 2.8. These types of images differ significantly from the ones used in this thesis.

Figure 2.8: Testing results with the corresponding bounding boxes displaying the location of smoke and fire, (Extracted from [7]).

2.5 Data Augmentation and Transfer Learning

The best way to make a model generalize better is to have more data to train it. In practice, there are limited amounts of data, sometimes even short datasets to train the models. One way to overcome this problem is to generate new data and add it to the training set. The different techniques that can be used to augment data will now be explored.

Some of the most basic transformations that can be applied to an image include flip (flip the image vertically or horizontally), rotation (rotate the image at any angle), scale (scale an image inward and outward), translation (move the image along the X-axis, Y-axis or both) and adding Gaussian noise (add noise to the image). Changing the brightness, saturation, hue, and contrast of an image can also be done to introduce even more variability. All these techniques can be combined and used to create new data and help enlarge the dataset.

2.5.1 Image-to-image translation

Neural style transfer proposed by Gatys *et al.* 2015 [35] is a technique that takes two images - a content image and a style image - and outputs a single image which is a combination of both. The output image

should be similar to the content image but ”painted” with a new style (e.g. Monet, Van Gogh). To perform this combination, two loss functions are defined: $L_{content}$, which measures the difference of content in two images, and L_{style}, which measures the difference in two images with relation to their style. Then, given the input (content image), the image is transformed, while minimizing $L_{content}$ and L_{style}. This technique produces realistic results but has a few limitations: for example, the style cannot be applied only to a certain part of the input image.

Introduced in Zhu *et al.* 2017 [36], the CycleGAN network presents a viable solution to the unpaired image-to-image translation problem. In most cases, image-to-image translation requires a dataset with pairs of images, e.g. the same image in summer and winter, if one wants to perform a summer-winter translation. However, building this kind of dataset is very difficult and sometimes even impossible (e.g. a painting could have no photo in the real world). To address this problem, CycleGAN networks can be used with unpaired images, meaning that the training dataset can be composed of a large collection of images in the summer and a large collection of different images in the winter. The CycleGAN is an extension of the GAN architecture, which includes two discriminators and two generators. One generator takes as input images from the first domain and outputs images for the second domain, while the other generator does the opposite. Each one of the discriminators checks whether the images produced by the generators are plausible in each one of the domains. The CycleGAN architecture uses cycle consistency: this means that the output produced by the first generator is fed as input to the second generator. The second generator produces an output image that should match the original image. To measure the differences between these two images, the network uses an additional loss measure - cycle consistency loss. In real-world applications, the CycleGAN network produces acceptable results, namely in style transfer (applying an artistic style to photographs), object transfiguration (being the most famous case converting a horse in a zebra and vice-versa) and season transfer (e.g. summer-winter translation), shown in Figure 2.9.

However, other more complex techniques have been developed in the last years, usually based on generative and adversarial approaches. The following subsections describe the most important ones.

2.5.2 Few/One-shot Learning

One-shot learning is a classification task where only one or a few examples are needed in order to learn the important information of each class and thus be able to classify new examples in the future. Whereas most deep learning architectures need a large training dataset to train the parameters of the network, one-shot learning aims to use only a few examples and be able to generalize well enough for each training example, to achieve good accuracy in newer images.

Introduced in Vinyals *et al.* 2016 [37], matching networks achieve higher accurate results in known datasets when compared to some of the best deep learning models. A support set *S* of *k* labelled

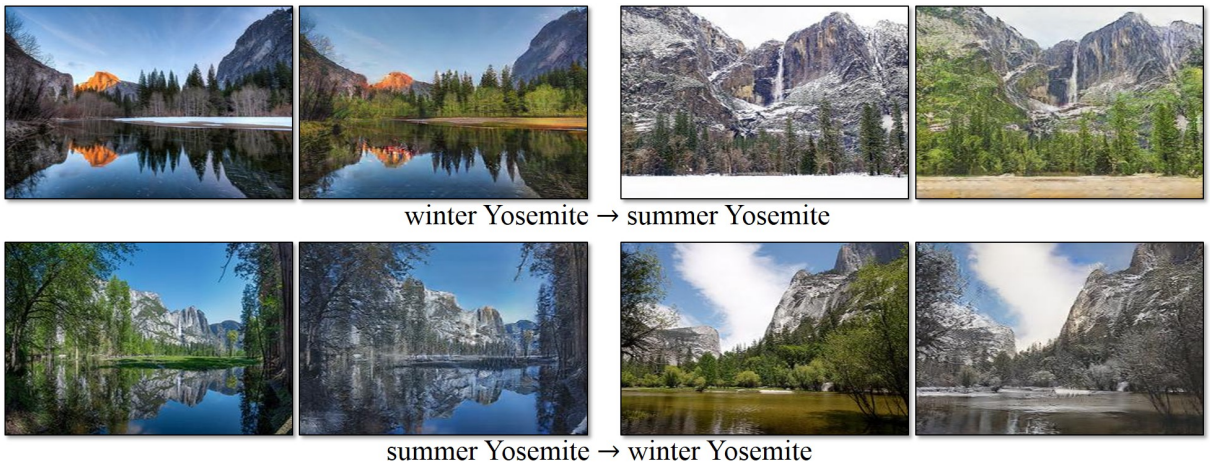

Figure 2.9: The use of CycleGANs to apply the season transfer effect, (Extracted from [36]).

examples, with input *x* and label *y* is shown to the matching network. Given a new example *x'*, the goal is to calculate the probability distribution and predict the correct label.

The calculation of the probability distribution relies on choosing an attention mechanism to measure the similarity between the set of examples in the support set *x* and the new example *x'*. The similarity is measured using the cosine distance and then passed to a softmax layer that maps the values between 0 and 1, where 0 represents the absence of similarity and 1 marks highly similar examples. To train the network, it is used a small number of labels (e.g., $\{cats,dogs\}$). From that, a support set *S* and a batch *B* are sampled (both *S* and *B* are labelled examples of cats and dogs). The network is then trained to minimise the error predicting the labels in the batch *B* conditioned to the support set *S*.

To evaluate the model, the authors have used three different variations of the ImageNet dataset. In two of those variations, they achieved better results compared to the baseline classifier (Inception). This model is fast to train, requires fewer training examples, and still achieves good performance. The biggest limitation is that does not perform so well when dealing with fine-grained images.

In [38], siamese neural networks are used for one-shot image recognition. In general, the image representations are learned via a supervised approach with siamese neural networks, and then the network is reused for one-shot learning.

The model was first trained on a subset of the Omniglot dataset (a set of handwritten characters). Once trained, the network is ready to perform one-shot learning. A test image *x* and some other images x_c (with x_c representing examples of each of the *C* classes) are given to the network. Then, the network, based on *x* and x_c, predicts the class corresponding to the maximum similarity.

The model was evaluated using a different alphabet than the one used for training and achieved a 92% accuracy. The model was then evaluated on the MNIST dataset, to see if the network was able to generalize well enough. However, it only achieved 70.3% of accuracy, which is far worse than state-

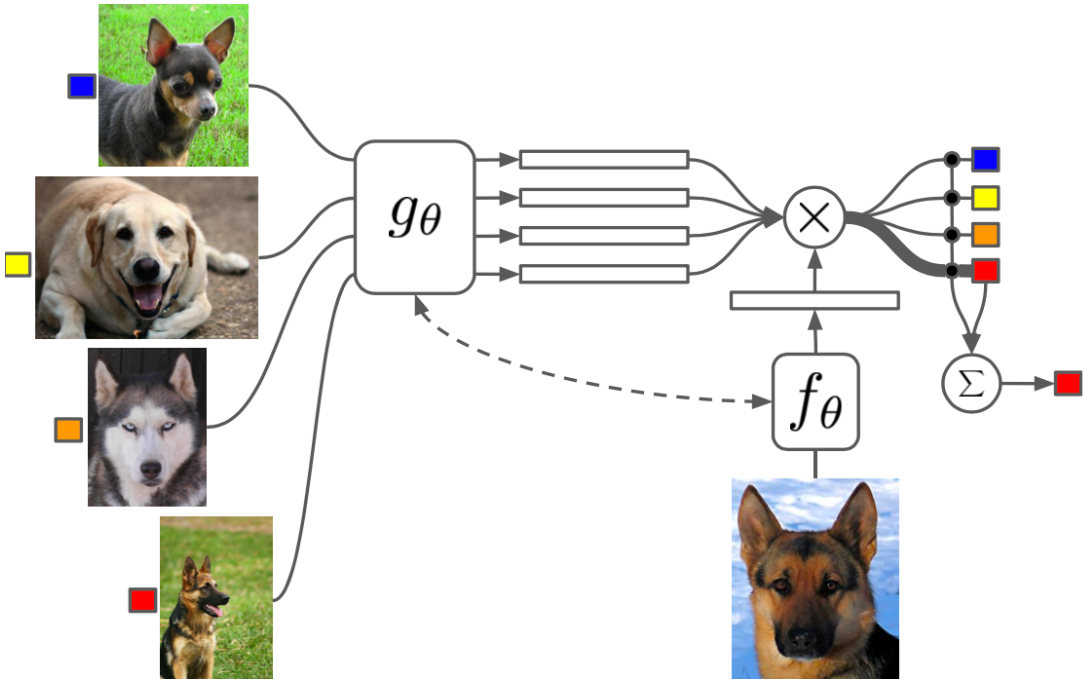

Figure 2.10: Matching Networks architecture, (Extracted from [37]).

of-the-art models can do. It can be inferred that this approach achieves viable results only when both datasets (train and evaluation) are not too different.

2.5.3 Transfer Learning

Sometimes, to develop complex deep learning models, large amounts of data are needed to train the model. However, getting vast structured and labelled data can be difficult and it is not always possible. A good example of a vast structured dataset is ImageNet [13], which contains more than 14 million hand-annotated images belonging to many categories. Transfer learning refers to the situation where the knowledge obtained after performing one task is stored and applied in a different problem. In the case that there is significantly more data in the first dataset (used to gain knowledge), then that knowledge may help to learn representations that are useful to generalize from just a few examples in the second dataset (the one to perform the classification task). This is often the case because images from different domains share low-level notions of edges, shapes, geometric changes, lighting change, etc.

In Raina *et al.* 2007 [39], the authors propose a new framework, "self-taught learning", for using unlabeled data in supervised classification tasks. The algorithm starts by using unlabeled data to learn a higher-level, more succinct, representation of the inputs. This makes the algorithm learn the most basic elements of an image, such as strong correlations between pixels, that may correspond to edges. After this process, applying this learned representation to the labelled data translates into better model

performance. The experiments made in this work used data from similar domains for the unlabelled and labelled data and achieved better performance compared to traditional machine learning classification algorithms.

In Wang *et al.* 2020 [40], the researchers propose attentive feature distillation and selection (AFDS), which help to adjust the strength of transfer learning regularization and also dynamically determine the important features to transfer. During transfer learning, they explore the output response of each convolutional layer in the source model when images from the target dataset are shown. Using this method, the fine-tuned model can still learn the behaviour of the source model and the weights in the target model can be optimized freely, increasing the generalization capacity. This method is referred to as attentive feature distillation (AFD) and it is used to learn which are the relevant features to transfer. To accelerate the transfer learning process, they propose attentive feature selection (AFS) to prune networks dynamically. AFS is designed to select important channels in the convolution to evaluate and skip irrelevant ones. Neurons that are rarely activated can be removed from the network, improving memory consumption. Using both AFS and AFD (AFDS), not only a higher target task accuracy is achieved (compared to existing pruning methods) but it can also allow a new model to learn faster by computing only a subset of channel neurons in each convolutional layer.

2.6 Summary

This chapter presented an overview of the related work relevant to this thesis. Firstly, some of the state-of-the-art deep learning models in image classification were discussed. These models are mainly based in CNNs with multiple layers to extract high-level features. Then, two types of networks that are able to generate new images were introduced: GANs and VAEs. Whilst there are some limitations to the quality of the generated images using VAEs, GANs offer a viable solution that is able to generate high-quality images. Concerning the limited data in some datasets, a couple of approaches, such as data augmentation and transfer learning, were discussed. Furthermore, a few works to detect fire or smoke were approached, mainly based on computer vision methods and convolutional neural networks. The majority of these works do not use datasets that represent early-stage fires, but rather evident images of flames or smoke.

3

Deep Neural Models for Image Classification and Generation

Contents

3.1 Image Pre-Processing 27
3.2 Deep Learning Classification Architectures 29
3.3 StyleGAN 34
3.4 Summary 37

Some of the models introduced in the previous section are still being widely used for image classification, like the VGGNet and the Inception architecture. However, several years have passed since their creation and new models with better performance and efficiency have been developed. At the same time, the best models that exist nowadays have high memory and processing requirements, as there are enough computational resources to develop and train those architectures. For instance, the ViT-G/14 architecture developed by Google researchers in 2021 [41] is able to reach the highest top-1 accuracy in the ImageNet dataset ever, at 90.45%. The computational power needed to train this network is also very demanding, as the ViT model has two billion parameters.

Regarding the image generation stage, and having the goal of improving any smoke/fire datasets, namely increasing the number of fire images, a couple of image generation techniques were employed. In this work, the StyleGAN [26] architecture was used for image generation since it is possible to fine-tune the model into a custom dataset. The BigGAN [28] model was not tested in this project because its developers do not publicly provide the discriminator: "We are releasing the pre-trained generator to allow our work to be verified, which is standard practice in academia. It does not include the discriminator to minimize the potential for exploitation", the authors refer.

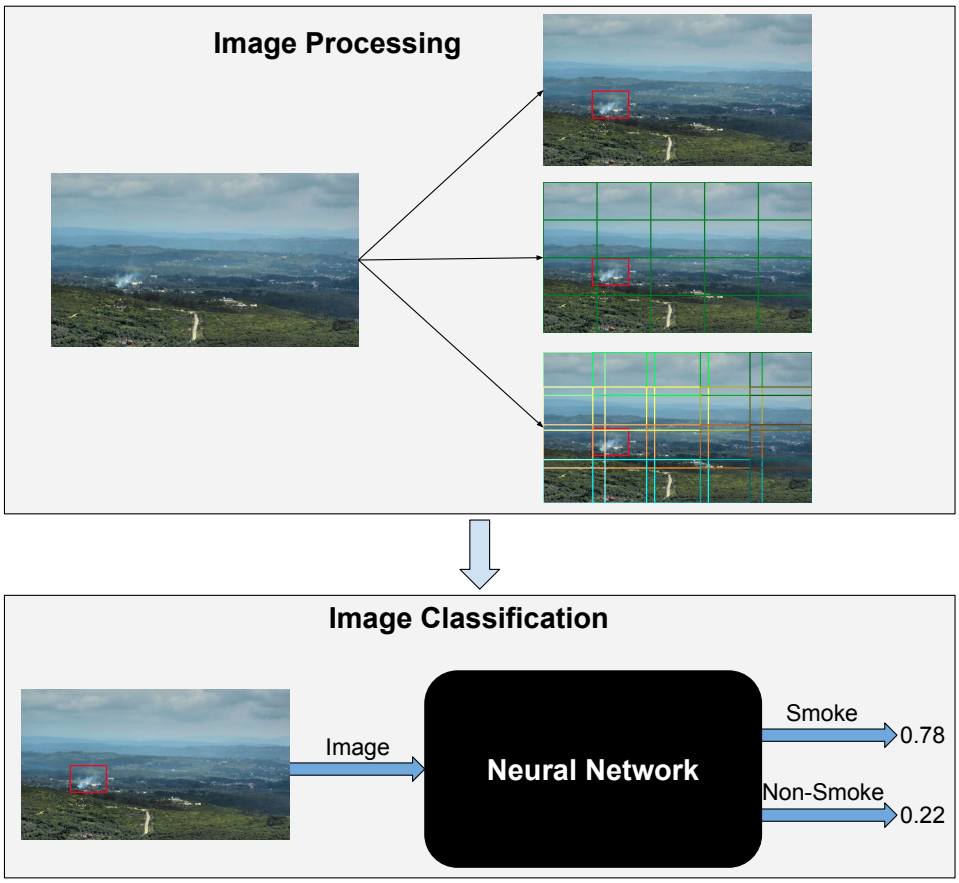

Figure 3.1: Overview of the proposed system.

The overall overview of the proposed approach is shown in Figure 3.1. First, the image processing step is executed, including the image subdivision. Then, the neural network is trained with smoke and non-smoke images. Finally, the neural network is ready to perform image classification: it receives an image as input and outputs the respective label, ”smoke” or ”non-smoke”.

This chapter will be divided as follows. Section 3.1 will detail the proposed approach used in the image pre-processing step, which includes the division of the image in several blocks and also the identification of the fire region in the image. This section will also explain the several methods used to create the train datasets used in both image generation and image classification networks. Section 3.2 will describe the deep learning architectures that were chosen to perform image classification: Xception, InceptionResNetV2 and EfficientNet. This section also includes the explanation of the fine-tuning phase, with the use of two transfer learning approaches. Finally, section 3.3 will describe the StyleGAN architecture that was used to generate new smoke images.

3.1 Image Pre-Processing

The dataset used in this work has both smoke and non-smoke images. For every smoke image, there is information that describes the number of smoke plumes present in the image and their location, which is given by 4 coordinates (Xmin, Ymin, Xmax, Ymax) that form the bounding box of the smoke region. In particular, (Xmin, Xmax) and (Ymin, Ymax) represent, respectively, the width and height of the bounding box. It was possible to calculate the area that a given fire occupies in the image, by dividing the area of the smoke's bounding box by the total area of the image. Using this information, the first step consisted in drawing a rectangular shape around the smoke region, as shown in Figure 3.2.

Figure 3.2: Smoke image example. A red rectangular shape displays the location of the smoke plume.

3.1.1 Image division

Dividing an image into blocks and performing the image classification step in each of these small blocks has a couple of advantages compared to classifying the entire image. Firstly, when classifying small portions of the image and later combining the results in a single output, the model only pays attention to one block at a time. Therefore, small objects that seem irrelevant in the overall picture can be more precisely detected when considering the block in which the object is inserted. Regarding the several datasets used in this work, the majority of images represent early-stage fires that occupy a small area in the image. Another advantage when considering classifying individual portions of the image is the ability to locate the position of the object. In the case of this thesis, the precise location of the fire is not a mandatory requirement but in real-world scenarios, it could be a useful feature. The major disadvantage when classifying small blocks in the image is the detection of false positives. For instance, small smoke plumes can be mistaken as clouds or fog when considering small areas in the image. Therefore, the model is not taking into account the environment around these phenomenons, making it more prone to signal false alarms. Also, if the model signals a false positive in a single block, the entire image is then labelled as having a fire. To study the potential advantages and disadvantages when classifying the entire image and classifying smaller parts of the image, the image was divided using two methods.

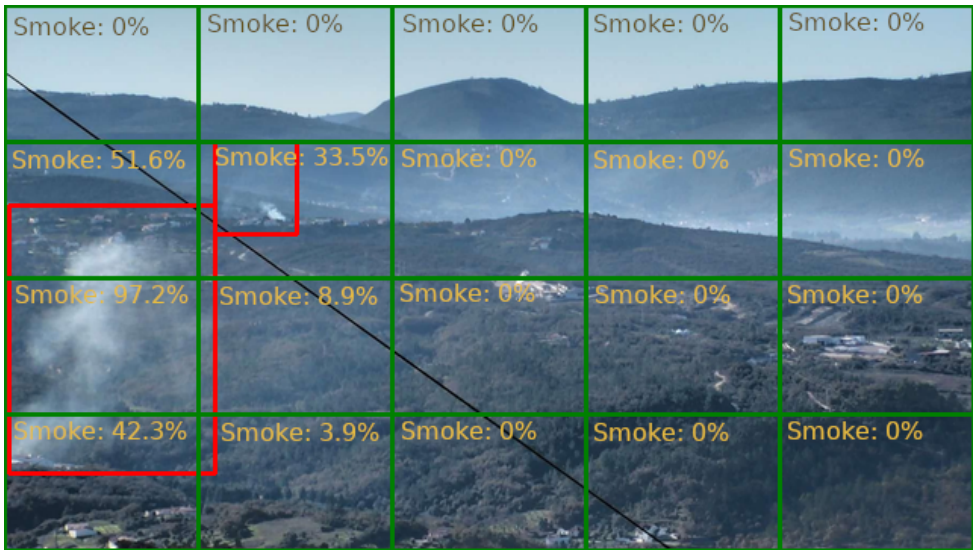

Figure 3.3: A 5x4 grid image with two hotspots outlined by a red rectangular shape. The smoke area in each grid is also displayed.

3.1.1.A Normal grids

The first approach consists in converting every smoke and non-smoke image into a $m \times n$ grid, where each individual grid has the same size. It is important that the individual blocks are not too small to avoid

potential false alarms, where the smoke plumes are mistaken as clouds, for example. Therefore, the image was divided into a 5×4 grid. For training purposes, it was important to calculate the smoke area in each of the individual grids. The image division process can be seen in Figure 3.3.

3.1.1.B Overlapped grids

One disadvantage in dividing the image into normal grids is that small smoke plumes can be present in the borders of consecutive grids, which may result in an overall loss of information. This can happen when the smoke location is spread across, for example, four neighbours (top right, top left, bottom right and bottom left), making each one of the four grids with a very small portion of the smoke. To overcome this problem, the second approach consists in having consecutive grids overlap each other in 15%, as it is shown in Figure 3.4. In this case, the image is also divided into a 5×4 grid but each grid is slightly bigger and contains information about its neighbours.

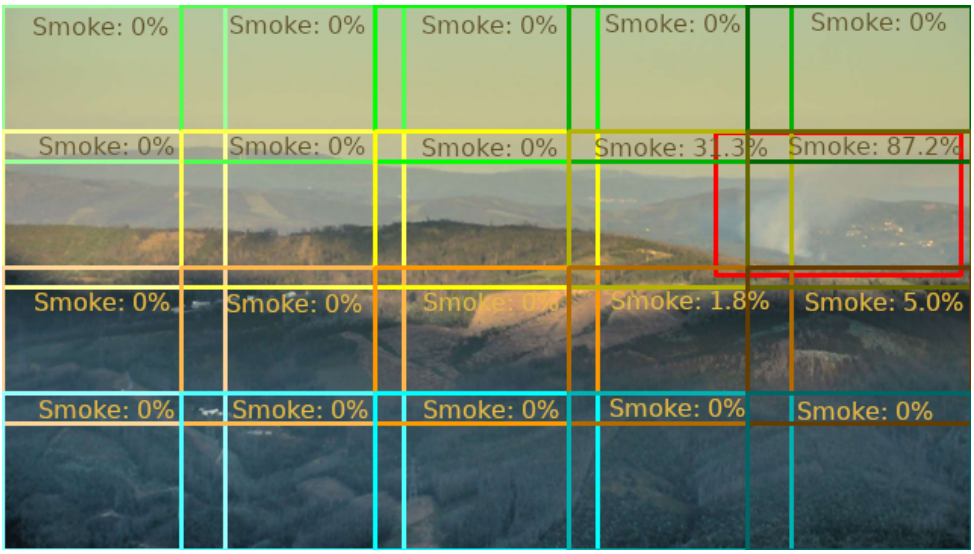

Figure 3.4: Sample snapshot from the grid overlap dataset. Each grid overlaps 15% to its neighbours. The smoke is outlined in a red rectangular shape.

In the evaluation phase, in order to label an image as smoke or non-smoke, all 20 grids are separately classified by the model. If the model output is greater than 0.5 in any of the grids, the global image is labelled as smoke. Otherwise, the output label is non-smoke.

3.2 Deep Learning Classification Architectures

The next subsections will detail the proposed framework that was implemented to perform image classification on smoke images. It will also describe the deep neural networks used for training and testing purposes.

3.2.1 Proposed framework

The main objective of this thesis is the early detection of wildfires during the surveillance of forest areas. The developed algorithm must be able to detect early-stage fires and simultaneously minimize the number of false alarms. To achieve this goal, three different deep CNNs were explored and two different methods to fine-tune the original networks were devised. Transfer learning has proven to be very effective in several classification tasks, as better results can be achieved with less training effort [42, 43], where a model is pre-trained in a larger dataset and then fine-tuned to perform a target task. In this work, we used the Keras API [44], which offers a vast number of pre-trained models in the ImageNet dataset. A few changes had to be made so that the model only supports the two classes of our domain: ”smoke” and ”non-smoke”. Two transfer learning approaches were developed and applied to each one of the original models.

In the first approach, designated **TN-1**, two fully connected layers were added to the models. The output of the original models had originally 2048 neurons, so the first layer added is a fully connected layer with a ReLU activation function to reduce the dimension of the vector, outputting a feature map with size (batch size, 32), where batch size corresponds to the number of batch images. The last fully connected layer added has a sigmoid activation function and outputs a single number, corresponding to the prediction of the input. If the output of the last layer is less than 0.50, the label is ”non-smoke”, otherwise the output is ”smoke”. In the fine-tuning phase using the custom dataset, the entire network was trained using a very small learning rate, lr=0.0001, in order to avoid potential loss of information. In this approach, the models were trained for 60 epochs in all datasets.

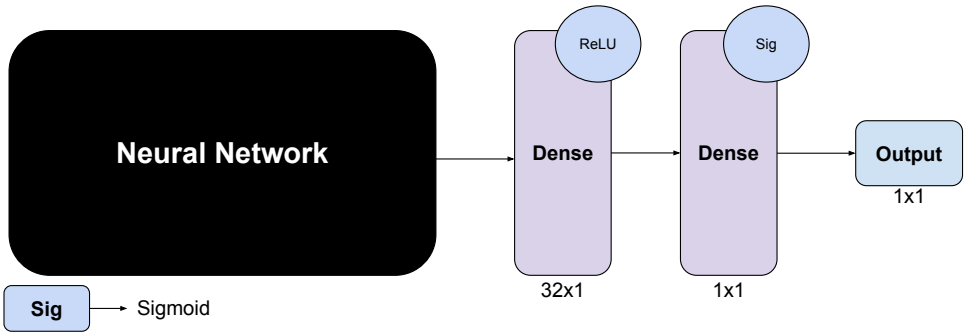

Figure 3.5: Deep learning architecture overview using the first transfer learning method, **TN-1**.

The second approach, designated **TN-2** and based in [45], was implemented to avoid the loss of the original information that is present in the pre-trained weights of the original model. First, the original layers are frozen, in order to avoid the destruction of the information they contain. Secondly, new, trainable layers were added on top of the frozen layers. In this case, four new layers were added. The first one is an average pooling layer, that reduces the dimension of the input feature map. It outputs a two-dimensional vector with size (batch size, channels), where batch size corresponds to the number of

batch images and channels is the number of channels in the input feature map (2048 in this case). The second added layer is a dropout layer, that helps prevent overfitting by randomly discarding some of the input units. In this case, the dropout ratio is 0.2, which means 20% of the input units will be discarded at each training step. The next layer is a fully connected layer with a ReLU activation function to reduce the dimension of the vector, outputting a feature map with size (batch size, 32). The last fully connected layer added has a sigmoid activation function and outputs a single number, corresponding to the prediction of the input. The network is then trained with a learning rate lr=0.001, and since the weights of the original networks are frozen, only the weights of the newly added layers are changed. The goal is that the newly added layers will learn to transform the old features into predictions in the new custom dataset. The last step is to unfreeze the layers from the original model and train the whole network with a very small learning rate (lr=0.00001). The goal is to slightly improve the whole network's accuracy by training it as a whole in the custom dataset while retaining the original information. Using this approach, the first and second training steps were both executed for 30 epochs.

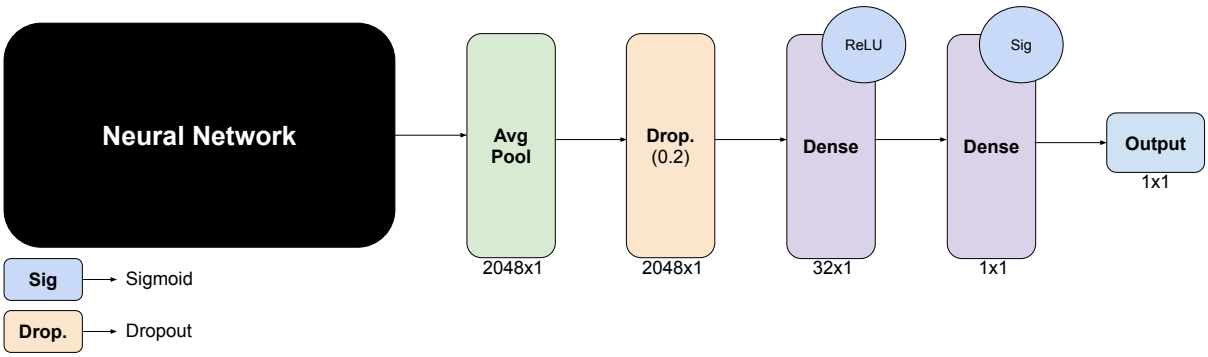

Figure 3.6: Deep learning architecture overview using the second transfer learning method, **TN-2**.

In both transfer learning methods, and after training the models, the weights corresponding to the best epoch related to the validation accuracy were saved to later proceed with the evaluation phase. Both methods were trained using the Adam optimization algorithm.

To compare the results in the evaluation phase, three different models were chosen. Since the Keras API offers more than 20 pre-trained models, the criteria used to choose the networks was the top-1 and top-5 performance in the ImageNet dataset. Therefore, the models that achieved the highest accuracy were the Xception [10], InceptionResNetV2 [11] and EfficientNetB6 [12] architectures.

3.2.2 Xception

Introduced in Chollet 2020 [10], Xception stands for an "extreme" version of the Inception architecture. The base of the Xception architecture is a modified version of the *depthwise separable convolution* operation. The original depthwise separable convolution is simply a depthwise convolution followed by

a pointwise convolution:

1. Depthwise convolution consists in applying an $n \times n$ convolution in each channel of the input. For example, if we have 4 channels, we apply a total of 4 $n \times n$ convolutions, one in each channel.

2. Pointwise convolution is performing a 1×1 convolution on the input in order to change its dimension (number of channels).

The "extreme" version of the Inception module has a key difference compared to the original Inception module: the order in which the operations are made. Firstly, a pointwise convolution is performed followed by a depthwise convolution. This important difference reduces the computational cost and memory needed to accomplish the whole operation compared to the original version. This happens because the number of $n \times n$ convolutions is smaller since we already reduced the input dimension in the pointwise (1×1) convolution.

The whole architecture can be seen in Figure 3.7 and it is composed of 14 modules, all of which have linear residual connections around them, except for the first and last modules. The input goes first through the entry flow, then is repeated eight times in the middle flow, and finally goes through the exit flow. The final number of parameters of the Xception network is 22 million, compared to the InceptionV3 23 million parameters and 138 million in the VGG16 model. In terms of performance, the Xception network is able to reach both higher top-1 and top-5 accuracies in the ImageNet dataset than VGG-16, ResNet-152 and InceptionV3 models.

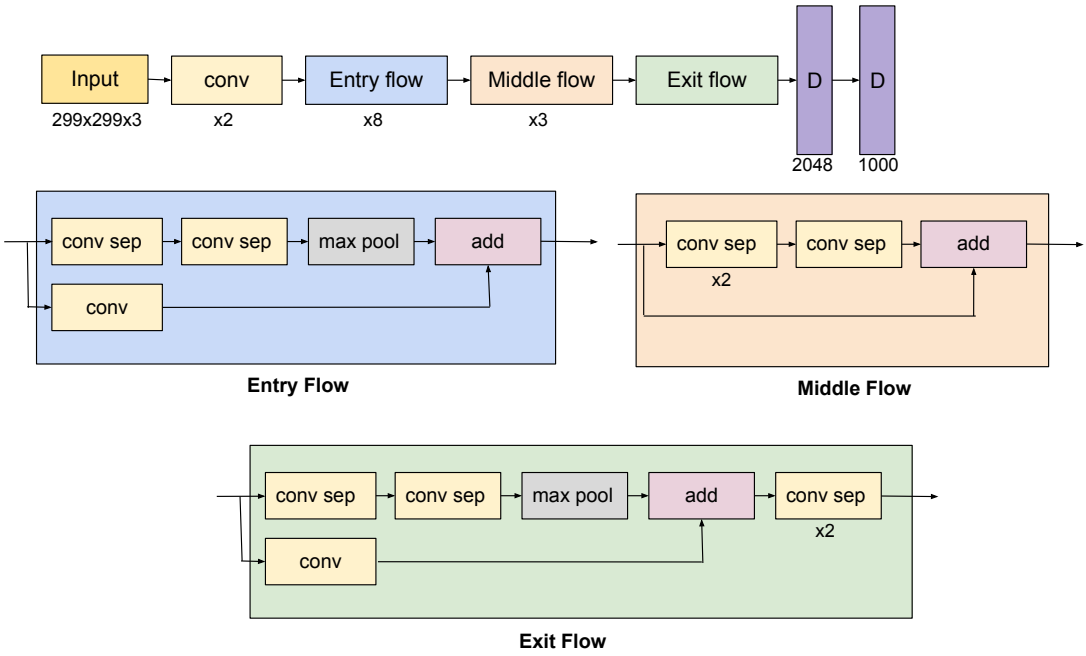

Figure 3.7: Xception architecture.

3.2.3 InceptionResNetV2

InceptionResNetV2 was introduced in Szegedy *et al.* 2016 [11] and is another variant of the original Inception network. In this network, multiple sized convolutional filters, that form the network modules, are combined with residual connections (Figure 3.8). The use of residual connections allows the creation of deeper networks mitigating the vanishing gradient problem, which may lead to better performance. This network is significantly deeper than the previous Xception architecture, having around 55 million parameters. The top-1 and top-5 accuracies in the ImageNet dataset are also slightly higher when compared to the Xception network.

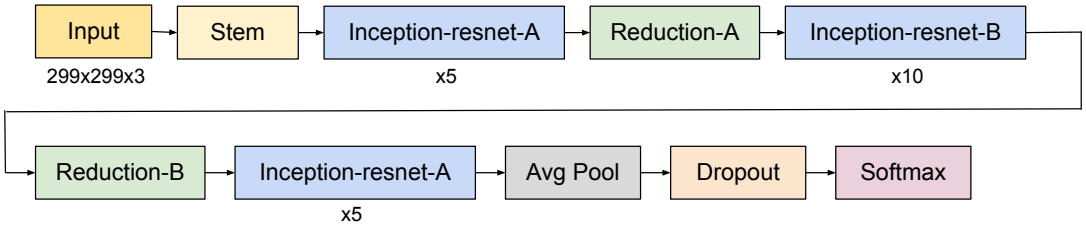

Figure 3.8: InceptionResNetV2 architecture.

3.2.4 EfficientNet

The EfficientNet network was introduced in Tan *et al.* 2019 [12] and is a convolutional neural network architecture that uniformly scales all dimensions of depth/width/resolution using a compound coefficient.

Increasing a network's depth is the most common way of scaling. As more layers are added, the intuition is that the network is able to capture richer and more complex features. However, deeper networks are also harder to train due to the vanishing gradient problem. One other way of scaling is through width and is commonly used for small size models. Wider networks tend to capture more fine-grained features, but not so well the higher-level features. In order to achieve higher accuracy, it is important to balance all dimensions of the network: width (w), depth (d) and resolution (r). The authors propose a new compound scaling method, using a compound coefficient Φ that scales all dimensions in a principled way:

$$\begin{aligned} depth : d &= \alpha^{\phi} \\ width : w &= \beta^{\phi} \\ resolution : r &= \gamma^{\phi} \\ s.t. \quad \alpha \times \beta^2 \times \gamma^2 &\approx 2 \quad and \quad \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3.1}$$

Starting from the baseline EfficientNet-B0, the first step is fixing $\phi = 1$. After the calculations, the best values for EfficientNet-B0 according to Equation 3.1 are $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$ (according to [12]).

In order to obtain EfficientNet-B1 to B7, α, β and γ are fixed as constants (with the values above) and the baseline network is scaled up with different values for Φ.

The experiments illustrated in Figure 3.9 compare the accuracy of the EfficientNet architectures with other state-of-the-art models in the ImageNet dataset. The results demonstrate that EfficientNet achieves better results compared to other models, whilst having significantly fewer parameters. In the testing phase of this work, EfficientNet-B6 was chosen due to its high accuracy and reasonable size: 43 million parameters.

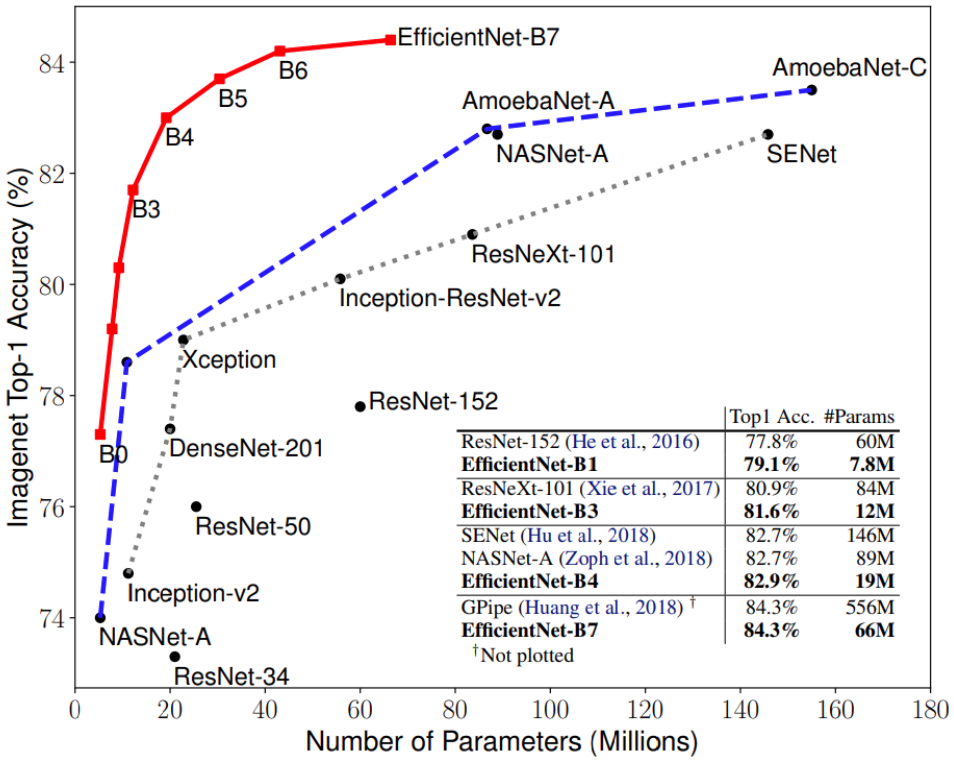

Figure 3.9: Model size compared to ImageNet accuracy in several architectures, (Extracted from [12]).

3.3 StyleGAN

One of the objectives of this thesis is also to explore new data augmentation techniques that could be used to improve the datasets related to smoke and fire. Despite traditional data augmentation techniques, such as image rotation, flip and scaling, are still being widely used when dealing with limited amounts of data, new image synthesis methods have been developed that can generate high-quality images. One of those methods is by using GANs, which are usually very unstable to train but can produce realistic results when it comes to image generation. In this work, the StyleGAN [26] architecture was used since it is possible to train it using a custom dataset and it also can generate high-quality images.

The StyleGAN network is an extension of the GAN architecture that allows controlling different style properties of generated images. The architecture of the generator is significantly different from regular GANs as the generator no longer takes a random vector from the latent space. The major differences in the architecture of the StyleGAN generator, shown in Figure 3.10, will now be covered.

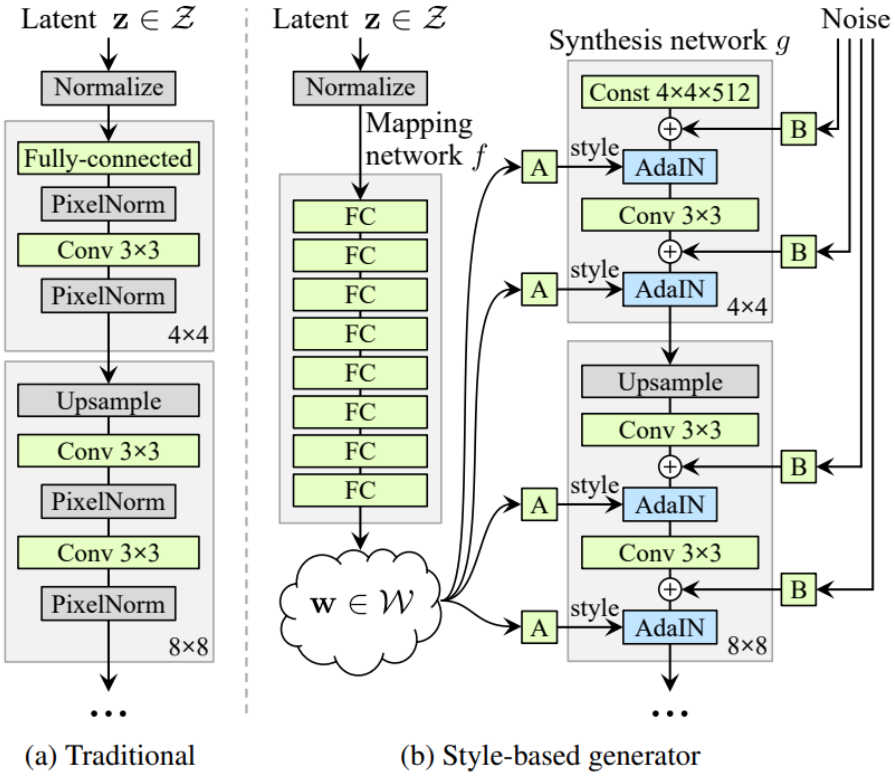

Figure 3.10: StyleGAN generator architecture, (Extracted from [26]).

StyleGAN uses a Baseline Progressive GAN and starts by training the network in smaller resolutions (4×4). Once both the generator and discriminator are stable, the width and height of the images is doubled (8×8) and the training process is repeated. For every new resolution, a new block is added to the synthesis network. This process is repeated until the desired output resolution is achieved, which is 1024×1024 in StyleGAN. Next, an independent mapping network is responsible for generating the style input vector, in which different elements control different visual features of an image. This network is composed of eight fully connected layers and outputs a 512 dimension vector ($4 \times 4 \times 512$). This style vector is then used in the synthesis network and, in each block, goes through a Adaptive Instance Normalization (AdaIN) operation that adds the style vector to the feature map. Before each AdaIN operation, a sample of Gaussian noise is added.

With all these changes, the StyleGAN network is not only able to generate high-resolution images but also very effective in controlling the several features of the image. For the experiments, the authors trained the network in both the Celeba-HQ and FFHQ datasets. The impressive quality of the generated

images can be seen in Figure 3.11.

Figure 3.11: Results produced by the generator with the FFHQ dataset, (Extracted from [26]).

The authors improve the original architecture of the SyleGAN architecture to create the StyleGAN2 version [46], which is able to improve the Fréchet Inception Distance (FID) score when generating new images. The most recent version of this model is named StyleGAN2-ADA [14]. The main difference from StyleGAN2-ADA to the original StyleGAN2 architecture is the use of adaptive discriminator augmentation, a technique that dynamically controls the augmentation strength based on the degree of overfitting. To test the potential use of smoke generated images to improve the size and quality of any dataset, StyleGAN2-ADA was used in this work. Two different approaches were tested in this architecture:

- Use of entire smoke images to train the GAN to try and generate full smoke images, in other words, images with small smoke hotspots and the corresponding landscape.
- Use of smoke hotspots to train the GAN in order to generate only new small smoke plumes. Then, these generated images were to be merged with normal images to create smoke images.

In order to generate new smoke images using StyleGAN, it was necessary to have many small smoke plumes images in the training dataset. Therefore, it was necessary to extract from the fire images only the smoke hotspots, based on the respective coordinates. The images were then resized to a 64×64

resolution. Figure 3.12 shows a few examples of smoke images that were created.

Figure 3.12: Examples of smoke hotspots extracted from fire images.

3.4 Summary

This chapter detailed the proposed approach to perform image classification on smoke images and also to generate new images that can be used to improve limited datasets. Section 3.1 described the pre-processing work that was implemented in smoke images. First, the smoke region was located in the respective image and the area that it occupies was calculated. Then, two types of image division methods were implemented. The first method divides an image into a 5×4 grid, where each block is independent of the others. In the second method, each block overlaps its neighbours, by sharing a small area of the image. Both of these methods were used to create the different datasets used to train and test the models. Section 3.2 detailed the proposed framework used to perform image classification. Two different transfer learning techniques were implemented to retrain the models in several training datasets. This section also detailed the three models chosen to perform image classification: Xception, InceptionResNetV2 and EfficientNet. Finally, section 3.3 details the StyleGAN architecture, used to

generate new images, and also the two different implemented approaches. The first approach includes the generation of full smoke images with the respective background and the second approach consists in generating only small smoke plumes to be later merged with normal images.

4

Dataset Description

Contents

4.1 INOV Dataset Description 40
4.2 Image Classification Datasets 42
4.3 Image Generation Datasets 43
4.4 Summary 43

4.1 INOV Dataset Description

The dataset used for training and testing purposes was provided by INOV [47] and has a total of 35328 RGB images with 1920×1080 resolution: 14125 represent images containing mainly smoke but also fire and the other 21203 are normal images with no smoke or fire. The first set includes images with different types of fires, from small to large smoke columns. For every fire in the dataset, there are different images of it, corresponding to different temporal snapshots of the same fire. While some fires are extinguished in the beginning and thus have few images, others take longer to disappear and have multiple snapshots. All of the dataset images were captured by surveillance cameras in Portuguese forests between 2019 and 2020, and represent the most diverse weather conditions: fog, clear sky, cloudy, etc. They were also taken at different times of the day (sunrise, sunset, night), which means they have very diverse lighting conditions. Figure 4.1 presents some examples of non-smoke images taken in diverse weather conditions.

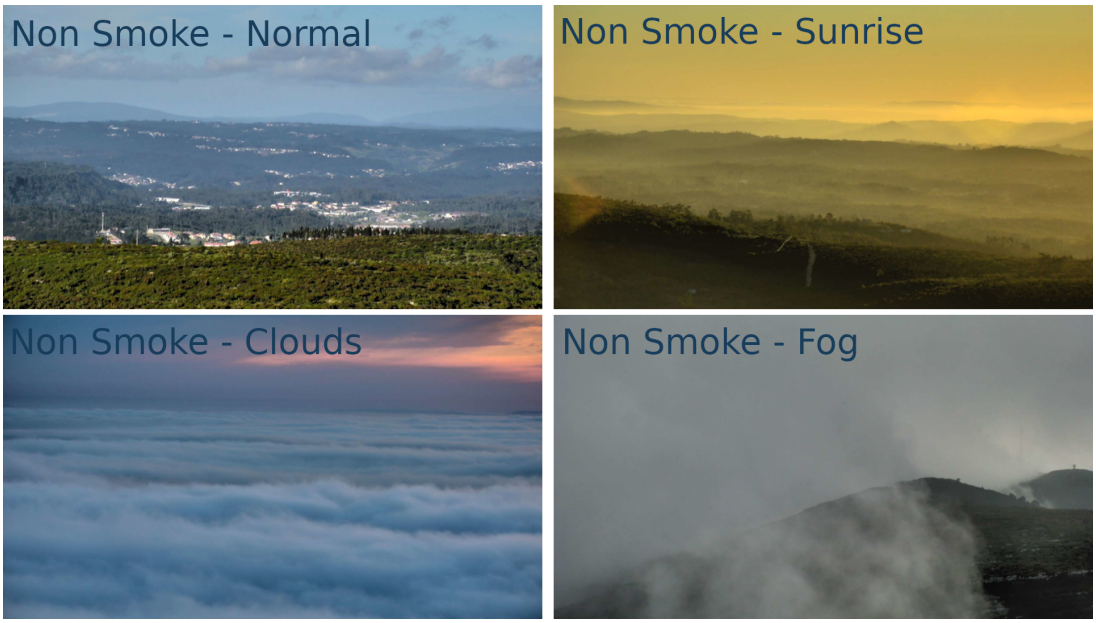

Figure 4.1: Examples of non-smoke images included in the dataset, captured in different conditions. The first example is a normal landscape. The second image is taken during sunrise. In the third image, only clouds can be observed. In the last image, a dense fog is present, severely limiting visibility.

The samples contained in the smoke dataset include images from a variety of fires that took place between 2019 and 2020 and include mainly white small smoke columns that correspond to early-stage fires. Some examples of the smoke images present in the dataset can be observed in Figure 4.2.

The images contained in the INOV dataset were captured by surveillance cameras installed in several towers and that are able to rotate 360º. The dataset is organized as follows:

1. Tower - The data and acquisition tower in which a given image was captured. There are a total of

Figure 4.2: Examples of smoke images included in the dataset.

9 towers that were used to capture the images present in this dataset.

2. Direction - Corresponds to the camera's direction at the time of acquiring the image. This value ranges from 0º to 359º.

3. Date - Corresponds to the day, month and year a given image was captured. In this case, the images were captured in the time period of 2019 and 2020.

Table 4.1 displays the distribution of the images in relation to the tower from which they were acquired.

Table 4.1: Distribution of Smoke (S) and Non-Smoke (NS) images by tower.

Tower id	**Frequency** (S)	**Percent**	**Frequency** (NS)	**Percent**
Tower 1	204	1.4	1942	9.2
Tower 2	2795	19.8	2920	13.8
Tower 3	474	3.4	3315	15.6
Tower 4	928	6.6	2393	11.3
Tower 5	2069	14.7	1304	6.2
Tower 6	3570	25.3	2329	11.0
Tower 7	2350	16.6	2409	11.4
Tower 8	1129	8.0	2822	13.3
Tower 9	606	4.3	1769	8.3
Total	14125	100	21203	100

4.2 Image Classification Datasets

4.2.1 Train Dataset

In order to train the CNNs, three different types of datasets were created. All datasets include both smoke and non-smoke images. Firstly, and in order to evaluate the impact that the number of images has in the training phase, two datasets (**DN-1** and **DN-2**) with different numbers of training images were built.

Two more datasets were created to study the impact of dividing the image into blocks and classifying these blocks individually. The second type of dataset created was the grid dataset (**DG**). This dataset is composed of grid images, using images from the method previously described in section 3.1.1.A. The overlap grid dataset (**DGO**) was created with images in which consecutive grids overlap each other, as described in section 3.1.1.B and exemplified in Figure 3.4. In both datasets, for training purposes, the grids that contain a smoke area greater than 5% are labelled as smoke. All other grids are labelled as ”non-smoke”.

All the images from the train datasets were scaled down from the original 1920×1080 resolution to 640×360. During the training phase, 80% of the images were used for training and the remaining 20% were used for validation purposes.

4.2.2 Test Dataset

The test dataset was created to evaluate the quality of the models after being trained and has a total of 1000 non-smoke images and 1000 images containing smoke. The test dataset was created after all training datasets and, to provide an unbiased evaluation, the images that compose the test dataset are not present in any of the training datasets. Table 4.2 provides an analysis of the smoke images present in the test dataset.

Table 4.2: Distribution of smoke images in the test dataset by smoke area.

Smoke Area (%)	Frequency	Percent
0-2	487	48.7
2-10	382	38.2
10-20	80	8
20-40	42	4.2
40-100	9	0.9
Total	1000	100

Since the objective of this work is to detect early-stage fires, it was important that the majority of images in the test dataset represent images with small smoke areas. As observed in Table 4.2, 86.9% of the smoke images in the test dataset have a smoke area less or equal to 10%. Medium size fires,

with a smoke area between 10% and 40%, represent 12.2% of the total smoke images. On the other hand, large size fires with a smoke area larger than 40% represent only 0.9%.

A summary of all datasets (train and test) is shown in Table 4.3. The sum of the number of images present in the training and test dataset was designed to be low when compared to the original number of the INOV. As some fires have multiple snapshots corresponding to similar images, it was important to choose for the training dataset only a few examples from each one, so that the model can generalize well enough. Regarding the test dataset, all images correspond to fires that were not used in the training dataset.

Table 4.3: Summary of the training datasets

Dataset Name	Smoke images	Non Smoke images
DN-1	320	360
DN-2	1840	2880
DG	6047	6047
DGO	6956	6956
Test	1000	1000

4.3 Image Generation Datasets

In order to generate images using the StyleGAN2 architecture, a high number of training examples need to be used to produce decent results. As previously mentioned in section 3.3, there are two types of train datasets: a dataset composed of normal smoke images (**DS**) and a dataset composed of extracted smoke hotspots (**DSH**).

The **DS** dataset is composed of all smoke images present in the INOV dataset with a total of 14125 smoke images, exemplified in Figure 4.2. For training purposes, all images were scaled down to 256×256.

The **DSH** has a total of 22255 training images. The number of smoke hotspot images is bigger than the number of overall smoke images because some fire images have multiple smoke plumes. All images from this dataset were also resized to a small 64×64 resolution because the majority of the smoke plumes represent a small area in the original image. A sample of images taken from this dataset is shown in Figure 4.3.

4.4 Summary

This chapter detailed the datasets that were used to train and evaluate the deep neural networks when performing image classification and image generation. Regarding image classification, two different types of datasets were created. The DN-1 and DN-2 datasets include normal images for both labels,

Figure 4.3: Examples of smoke hotspots from the DSH dataset, used to train the StyleGAN2-ADA network.

”smoke” and ”non-smoke”, where the only difference is that the DN-2 dataset has a larger number of training images compared to the DN-1 dataset. The second dataset type is the grid category that also includes two datasets, DG and DGO. The DG dataset is composed of individual blocks, where an image is divided into a 5×4 grid. In the DGO dataset, the only difference is that each grid overlaps its neighbours by 15%. The test dataset created to evaluate the image classification models includes different images from the ones present in the training datasets and was made to match the main goal of this thesis: the detection of early-stage fires. Regarding image generation, two types of datasets were created. Whilst the first DS dataset is composed of normal images, the DSH dataset is made of small smoke plumes that were extracted from the original smoke images.

5

Experimental Evaluation

Contents

5.1 Image Generation 46

5.2 Image Classification 51

5.3 Discussion 56

This chapter will detail the experimental evaluation of the proposed deep learning approach. Section 5.1 describes the experimental methodology and results achieved when generating new smoke images. Section 5.2 presents the results when performing image classification in the several training datasets using the proposed architectures and transfer learning methods. Finally, section 5.3 summarizes the results that were obtained and compares them with other papers with similar goals.

Both the image generation and image classification experiments were run using Google's cloud services [48]. The specifications of the virtual machine used include 2 Virtual Centralized Processing Units (vCPUs), 16GB RAM, a Nvidia Tesla T4 GPU with 16GB dedicated VRAM, and Ubuntu 20.04 as the operating system. This chapter is divided as follows. Section 5.1 will describe the methodology and results achieved in the image generation stage. Section 5.2 details the results achieved when classifying the smoke and non-smoke images. Finally, section 5.3 will compare the results achieved in the image classification step with other similar works regarding smoke and fire detection.

5.1 Image Generation

The image generation evaluation was performed using the StyleGAN2-ADA [14] architecture available in its official repository [49]. The image generation stage is divided into two main purposes: generating an entire smoke image and generating only small smoke hotspots. The latter is intended to be artificially merged with normal images to create smoke images. This section is divided into four main topics: subsection 5.1.1 details the evaluation metric used to assess the quality of the generated images; subsection 5.1.2 will present some results from the StyleGAN2-ADA authors, to have a base comparison model; section 5.1.3 will display the results that were achieved when generating full smoke images; finally, subsection 5.1.4 will describe the results achieved when generating small smoke hotspot images.

5.1.1 Metric

When using several methods to generate fake images, such as GANs or VAEs, it is important to have a measure that indicates how good the generated images are compared to real-world examples. One of the most popular metrics is the FID metric.

Heusel *et al.* 2017 [50] proposed the FID metric that relies on a pre-trained Inception-v3 network on ImageNet to capture the features of the images. It uses the last pooling layer before the classification output to capture the most relevant features of an input image. This method is done using a collection of real and generated images. The data distribution of these features is then modelled as a multivariate Gaussian distribution. The FID score between real and generated images is calculated using the two distributions (real and generated images distributions) with the Wasserstein-2 distance. A lower FID score represents a better quality image. The FID score is calculated using the following equation, where

μ_1 and μ_2 refers to the feature-wise mean of the real and generated images; Σ_1 and Σ_2 refer to the covariance matrix for the feature vectors of both images; and Tr refers to the sum of the elements along the main diagonal of the matrix:

$$FID = \|\mu_1 - \mu_2\|^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^2) \tag{5.1}$$

In practice, the FID metric works well when adding some kinds of distortions to an image, such as blur or noise, as the value will increase as more distortion is applied.

5.1.2 StyleGAN2-ADA

The main results achieved using the StyleGAN2-ADA architecture used two large datasets: the Flickr-Faces-HQ (FFHQ) dataset [51] and the LSUN cat dataset [52]. The FFHQ is a high-quality image dataset of human faces, originally created as a benchmark for the StyleGAN architecture. The dataset consists of 70,000 high-quality PNG images at 1024×1024 resolution and contains considerable variation in terms of age, ethnicity and image background. It also has good coverage of accessories such as eyeglasses, sunglasses, hats, etc. On the other hand, the LSUN cat dataset has a total of 1,657,266 cat images. To evaluate the quality of the generated images, the FID metric was used, where lower values translate into higher quality generated images.

Table 5.1: Comparison of the FID metric between the StyleGAN2 and StyleGAN2-ADA architectures in both FFHQ and LSUN cat datasets, using different dataset sizes during training.

Dataset	Size	**StyleGAN2** (FID)	**StyleGAN2-ADA** (FID)
FFHQ 256x256	1k	100.16	**21.29**
	5k	49.68	**10.96**
	10k	30.74	**8.13**
	30k	12.31	**5.46**
	70k	5.28	**4.30**
	140k	**3.71**	3.81
LSUN cat 256x256	1k	186.91	**43.25**
	5k	96.44	**16.95**
	10k	50.66	**13.13**
	30k	15.90	**10.50**
	100k	**8.56**	9.26
	200k	**7.98**	9.22

In practice, the use of StyleGAN2-ADA compared to the baseline StyleGAN2 allows achieving lower FID values when training datasets with limited data. The results presented by the authors are significantly better when the training dataset has less than 30k images. Table 5.1 compares the FID results from both SyleGAN2 and StyleGAN2-ADA achieved by the authors [14] using different training dataset sizes. Figure 5.1 displays a few examples of the generated images using the FFHQ and LSUN datasets for training.

Figure 5.1: Examples of generated images using the StyleGAN2-ADA architecture. The left figure represents generated cat images. The right figure has real images from the AFHQ Cat dataset [53] used for training. (Extracted from [14]).

5.1.3 Smoke Images

The first approach when generating new images was trying to generate smoke images with a similar forest background, just as in the INOV dataset. Therefore, the dataset used to train the StyleGAN2-ADA model was the **DS** dataset, previously described in section 4.3, containing 14125 smoke images. The StyleGAN2-ADA model allows to stop and resume training when desired. To do so, the model saves periodic snapshots that preserve the training information. In this case, the gap between saved snapshots is when the discriminator sees about 40k images, meaning the generator already generated this amount of images. The way to evaluate the quality of the generated images is by using the FID metric, the same method used by the StyleGAN2-ADA authors in their experiments. The results of training StyleGAN2 using the entire full smoke images dataset are summarized in Table 5.2.

Table 5.2: Evaluation results using StyleGAN2 using the full smoke **DS** dataset for training. #images [K] represent the number of images used during training, divided by 1000.

Snapshot	#images [K]	Time (hh:mm:ss)	FID (256x256)
1	40.4	00:17:30	336.47
2	80.7	01:27:04	327.77
3	121.1	02:26:57	142.45
4	161.5	03:37:59	71.77
5	201.8	04:49:53	52.57
6	242.1	05:58:40	46.41
7	282.4	07:08:02	42.67
8	322.8	08:17:25	41.16
9	363.1	09:25:05	**41.15**

It is important to note that the dataset used in this experiment is very different from the FFHQ, LSUN and even the AFHQ cat datasets used by the StyleGAN2-ADA authors. All these datasets offer high-quality images of close-range objects, whilst the **DS** dataset represent smoke images in the most diverse conditions, where the smoke region, in the vast majority of the cases, represents only a small portion of the image. Also, to establish a base reference, in order to obtain a FID value less than 5 in the FFHQ dataset with 140k 1024×1024 images, the authors trained the model for more than 5 days, using 8 parallel NVIDIA Tesla V100 GPUs.

In our experiments, the FID score started to stabilize around snapshot 6, when the discriminator had seen around 242k images, achieving the lowest score at 41.15. In total, the model was trained for 9 hours, 25 minutes and 5 seconds using 14125 images and the discriminator saw 363k images. This value is similar to the one achieved by the authors when training the LSUN cat dataset with 1k images. Figure 5.2 displays a few examples of the total 1k generated images.

In all images it is possible to notice a lot of green tones, representing trees and other vegetation and also a clear division between the sky and the rest of the background. A few images also include residential areas, in which it is possible to see white walls and red roofs. Finally, there are some images that contain areas resembling smoke plumes, despite being a little blurry at times. With these experiments, despite some of the images being very similar to real ones, it was not possible to include them in the smoke dataset, since the majority of the generated images did not have any smoke area.

5.1.4 Smoke Plumes Images

The second approach when generating new images was trying to generate only small smoke plumes with no background. Therefore, the dataset used in the training stage was the **DSH** dataset, previously described in section 4.3, containing 22255 smoke plumes images. In this case, the saved snapshots have a gap that corresponds to the discriminator having seen about 41k images. The results of training StyleGAN2 using the **DSH** dataset are summarized in Table 5.3.

Table 5.3: Evaluation results using StyleGAN2 using the smoke plumes **DSH** dataset for training. #images [K] represent the number of images used during training, divided by 1000.

Snapshot	#images [K]	Time (hh:mm:ss)	FID (64x64)
1	41.1	00:38:02	236.99
2	82.0	01:09:29	131.44
3	123.0	01:40:53	60.45
4	164.0	02:12:18	42.32
5	204.9	02:43:41	28.92
6	245.9	03:15:07	20.90
7	286.8	03:46:34	14.84
8	326.9	04:19:26	**12.61**

Figure 5.2: Examples of 256×256 full smoke generated images using the StyleGAN2-ADA architecture.

With these experiments, the best FID score achieved was 12.61 in snapshot 8, when the discriminator had seen 326k images. Comparing both approaches, we can conclude that the FID value was lower in this second method, which allows generating images with higher quality. A possible explanation is the complexity of the training images: whilst in the first **DS** dataset, the images are complex with lots of different backgrounds and smoke plumes sizes, the second **DSH** dataset only contains small 64×64 smoke plumes. The best FID value (12.61) is similar to the one achieved by the StyleGAN2-ADA authors using the FFHQ 5k dataset and also the LSUN 10k dataset for training. In Figure 5.3, we can see some examples of the total 1k generated smoke plumes images.

Figure 5.3: Examples of 64×64 smoke plumes generated images using the StyleGAN2-ADA architecture.

5.2 Image Classification

This section will present the results achieved when performing image classification in the several training datasets using the Xception, InceptionResNetV2 and EfficientNet architectures. The first and second sets of results are related, respectively, to the first and second transfer learning methods, **TN-1** and **TN-2**, previously described in section 3.2.1. The metrics used to test the model are described in section 5.2.1.

5.2.1 Metrics

In order to detail the relevant metrics, and considering the problem at hand, only 2 classes will be considered: Smoke and Non-Smoke. The Smoke class represents an image with smoke, flames, or any presence of fire. The Non-Smoke class corresponds to normal images. There are 4 important concepts to take into account:

- **True Positives (TP)** - The cases in which the model predicted SMOKE and the actual label is SMOKE.
- **True Negatives (TN)** - The cases in which the model predicted NON-SMOKE and the actual label

is NON-SMOKE.

- **False Positives (FP)** - The cases in which the model predicted SMOKE and the actual label is NON-SMOKE.
- **False Negatives (FN)** - The cases in which the model predicted NON-SMOKE and the actual label is SMOKE.

The most important metrics that can be used to evaluate image classification models are the following:

- **Accuracy** - accuracy is the number of correct predictions made by the model compared to all predictions that were made. It is a good metric to use when the variables in the target classes are nearly balanced. In this thesis, the majority of the training datasets have a balanced number of images in both classes.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \tag{5.2}$$

- **Precision** - precision is a measure that shows the proportion of images that were classified as having smoke, actually had smoke. This metric allows seeing if the number of false positives is too high compared to the number of true positives.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.3}$$

- **Recall** - recall is a measure that tells the proportion of images that actually have smoke, compared to the images that the model predicted as having smoke.

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.4}$$

- **F1 Score** - combines both precision and recall in a single value, by taking their harmonic mean. A good F1 score translates in both a low number of both false positives and false negatives, meaning that the model is correctly identifying real threats.

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

5.2.2 Experiments

The experiments performed using the first transfer learning method, **TN-1**, are summarized in Table 5.4. In this approach, a few layers are added at the end of the models in order to support the only 2 classes

and, then, the whole network is trained using the target dataset and a very small learning rate.

Table 5.4: Results for the TN-1 method.

Model	Dataset	Accuracy	Precision	Recall	F1
Xcep	DN-1	0.867	0.814	0.950	0.877
	DN-2	0.950	**0.975**	0.923	0.948
	DG	0.868	0.800	**0.981**	0.881
	DGO	0.917	0.897	0.942	0.919
IRNv2	DN-1	0.839	0.822	0.865	0.843
	DN-2	**0.954**	0.968	0.939	**0.953**
	DG	0.837	0.854	0.812	0.832
	DGO	0.833	0.843	0.819	0.831
EN-B6	DN-1	0.791	0.892	0.663	0.761
	DN-2	0.911	0.951	0.868	0.907
	DG	0.685	0.894	0.420	0.571
	DGO	0.653	0.599	0.927	0.724

When training all models with the first transfer learning method, the InceptionResNetV2 architecture presented the best accuracy (0.954) and F1 score (0.953) metrics trained in the **DN-2** dataset. The Xception model has the best results both in precision (0.975), when trained in the **DN-2** dataset, and recall (0.981), when trained in the **DG** dataset. The results show that the EfficientNet network reached lower values in most of the metrics compared to the other two models when trained in any of the datasets. Comparing both methods when classifying the entire image (datasets **DN-1** and **DN-2**) or individual image grids (datasets **DG** and **DGO**), the results were better when the network is trained and evaluated with the whole images. It is also important to note that the results of training both **DN-1** and **DN-2** datasets, where the only difference is the number of training images, show that a higher number of training images is better.

Table 5.5: Results for the TN-2 method.

Model	Dataset	Accuracy	Precision	Recall	F1
Xcep	DN-1	0.870	0.836	0.920	0.876
	DN-2	0.968	0.976	0.959	0.967
	DG	0.849	0.978	0.714	0.825
	DGO	0.860	0.980	0.734	0.839
IRNv2	DN-1	0.903	0.879	0.934	0.906
	DN-2	0.959	0.977	0.940	0.958
	DG	0.726	**0.993**	0.455	0.624
	DGO	0.778	0.984	0.565	0.718
EN-B6	DN-1	0.900	0.887	0.918	0.902
	DN-2	**0.982**	0.987	**0.977**	**0.982**
	DG	0.501	0.506	0.081	0.140
	DGO	0.505	0.739	0.017	0.033

The second set of results, regarding the second transfer learning method, **TN-2**, are shown in Table 5.5. In this approach, the training is executed in two phases: firstly we freeze the original network

and train only the recently added layers in the new dataset; secondly, the original network is unfrozen and the whole network is trained using a small learning rate.

When it comes to the second transfer learning approach, the EfficientNet-B6 network displays the best accuracy (0.982), recall (0.977) and F1 score (0.982), when trained in the **DN-2** dataset. Regarding the precision metric, despite the InceptionResNetV2 model having the best value (0.993), all other metrics are significantly lower compared to the best EfficientNet-B6 result. Similar conclusions to the ones in the previous method can be drawn when comparing both normal and grid datasets: both normal datasets achieved better results in all models when compared to the grid datasets. Despite having the overall best result, the EfficientNet network was unable to deal with both **DG** and **DGO** datasets, displaying in both poor results, namely in the precision and F1 score. The cause of these poor results, for example in the **DG** dataset, is the high number of false negatives and true negatives, which combined resulted in 1840 ”non-smoke” predictions when the real number of ”non-smoke” images in the test dataset is 1000.

When comparing both transfer learning techniques, **TN-1** and **TN-2**, we conclude that the results for the normal datasets were better using the **TN-2** method. On the other hand, when considering the grid datasets, the results were better using the **TN-1** method. Overall, the best result is the one achieved in the second transfer learning approach, by the EfficientNet-B6 network.

5.2.3 Best Results Analysis

To better understand the obtained results, a deeper analysis was performed using the EfficientNet-B6 model trained in the **DN-2** dataset using the **TN-2** approach. Table 5.6 displays the total number of TP, TN, FP and FN classified by the model. With 98.2% accuracy, the model only predicted ”smoke” in a ”non-smoke” image in 13 out of the 1000 ”non-smoke” images. On the other hand, the model predicted ”non-smoke” in a ”smoke” picture in 23 out of the 1000 ”smoke” images.

Table 5.6: Total number of TP, TN, FP and FN.

True Positives	True Negatives	False Positives	False Negatives
977	987	13	23

Figure 5.4 displays a line chart of the train and validation accuracy using the **TN-2** transfer learning method. In the first training stage, where the original weights of the models are frozen and only the newly added layers are trained, the training accuracy is steadily increasing. The validation accuracy is also increasing, despite having some ups and downs. In the second training stage, where the whole network is trained, we conclude that the validation accuracy has a great improvement, getting closer to the training accuracy. We can also conclude that the model is not overfitting because the final train and validation have similar accuracy values, with a small 0.02 gap.

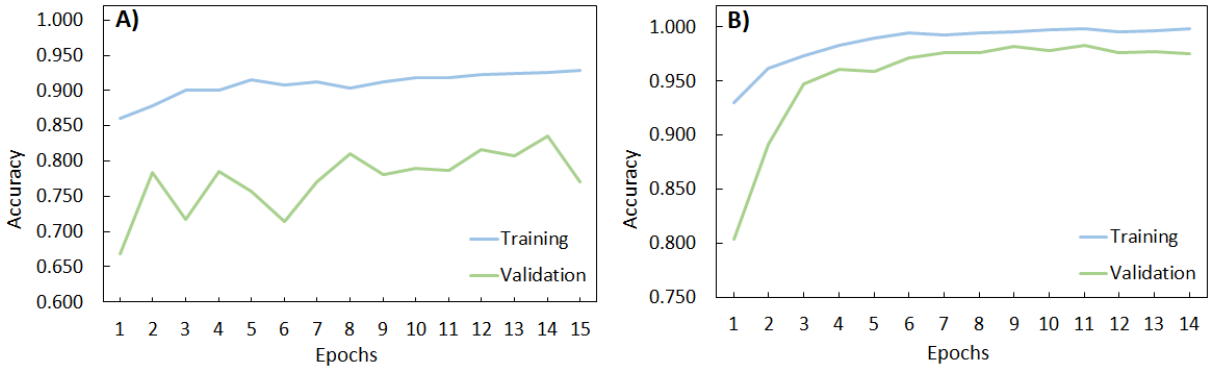

Figure 5.4: Charts **A)** and **B)** represent, respectively, the training and validation accuracy in the first and second training stages using the **TN-2** method.

To further analyse the results, a visual inspection of the images was performed. To highlight this analysis, the following images are annotated with a corresponding label (1 for ”smoke” and 0 for non-smoke) and the corresponding prediction. For this reason, if the label is 1 (smoke), a prediction value closer to 1 indicates that the model is very confident about the prediction. On the other hand, if the label is 0 (non-smoke), a prediction value closer to 0 shows more confidence by the model.

The first examples are images that the model classified as ”smoke” but they actually do not contain smoke, in other words, false positives. The majority of false positives correspond to images that either have fog or clouds or images that are a little blurry, as shown in Figure 5.5.

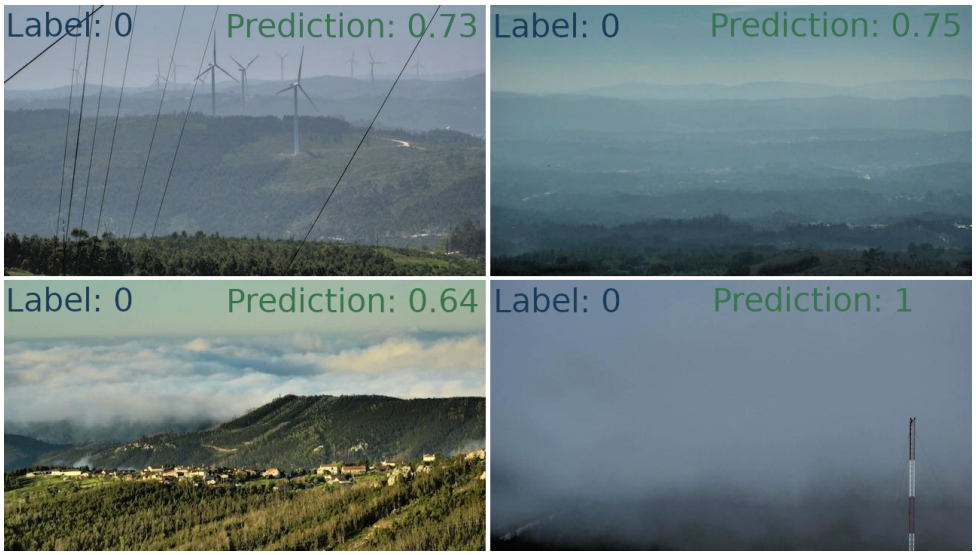

Figure 5.5: Examples of False Positives.

Figure 5.6 shows examples of true positives, in other words, images that had smoke and were correctly labelled as ”smoke”. The majority of the examples presented show small smoke columns, which the model correctly identified.

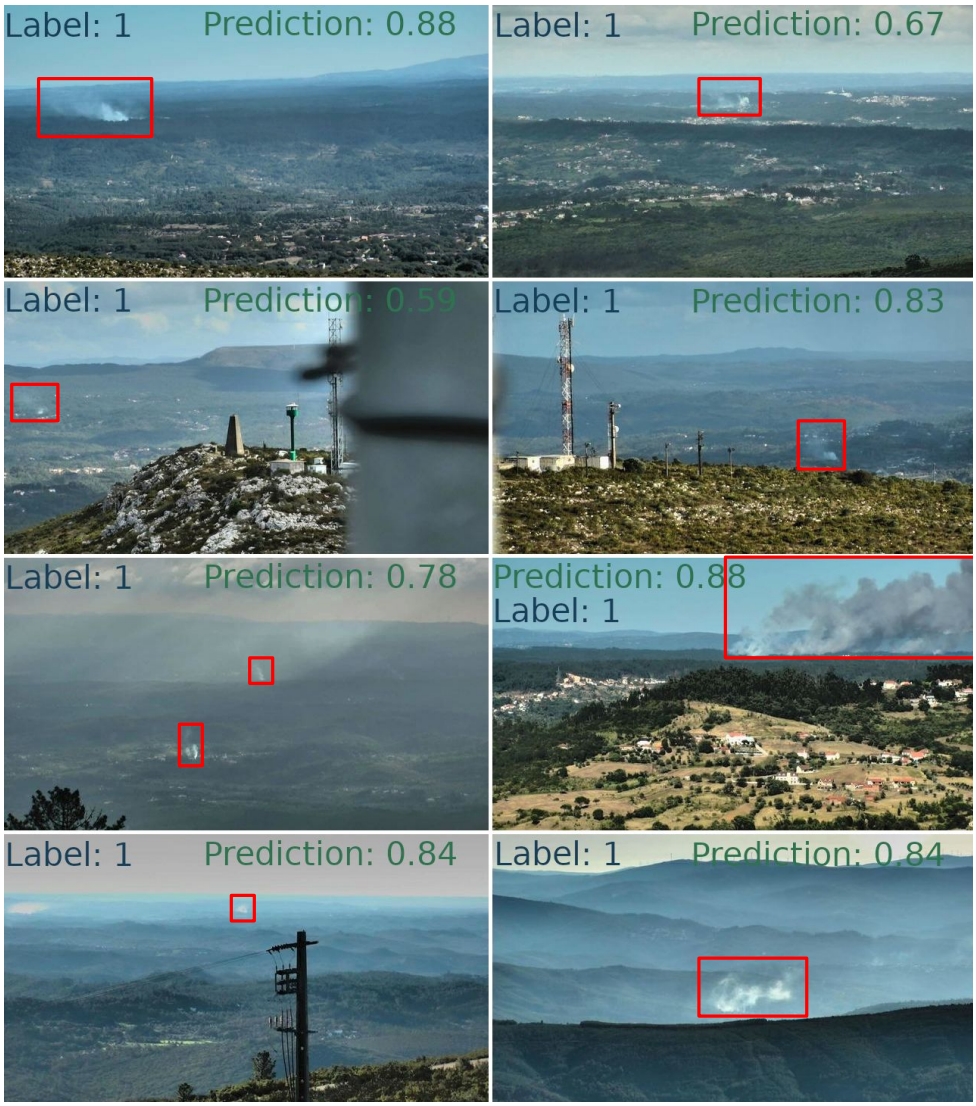

Figure 5.6: Examples of True Positives. A red bounding box identifies the smoke location.

The last few examples, displayed in Figure 5.7, are regular images that do not contain smoke and the model also correctly classified them as ”non-smoke”. The first three examples are particularly interesting as they have many clouds that could be interpreted as smoke.

5.3 Discussion

This section compares the results that were achieved in this thesis to other results achieved in similar works, summarized in Table 5.7. The majority of the related works focuses on detecting large areas of fire or smoke in images, not considering the importance of detecting these incidents at an early stage.

Jiao *et al.* 2019 [7] proposes the use of the YOLOv3 object detection architecture to train the model

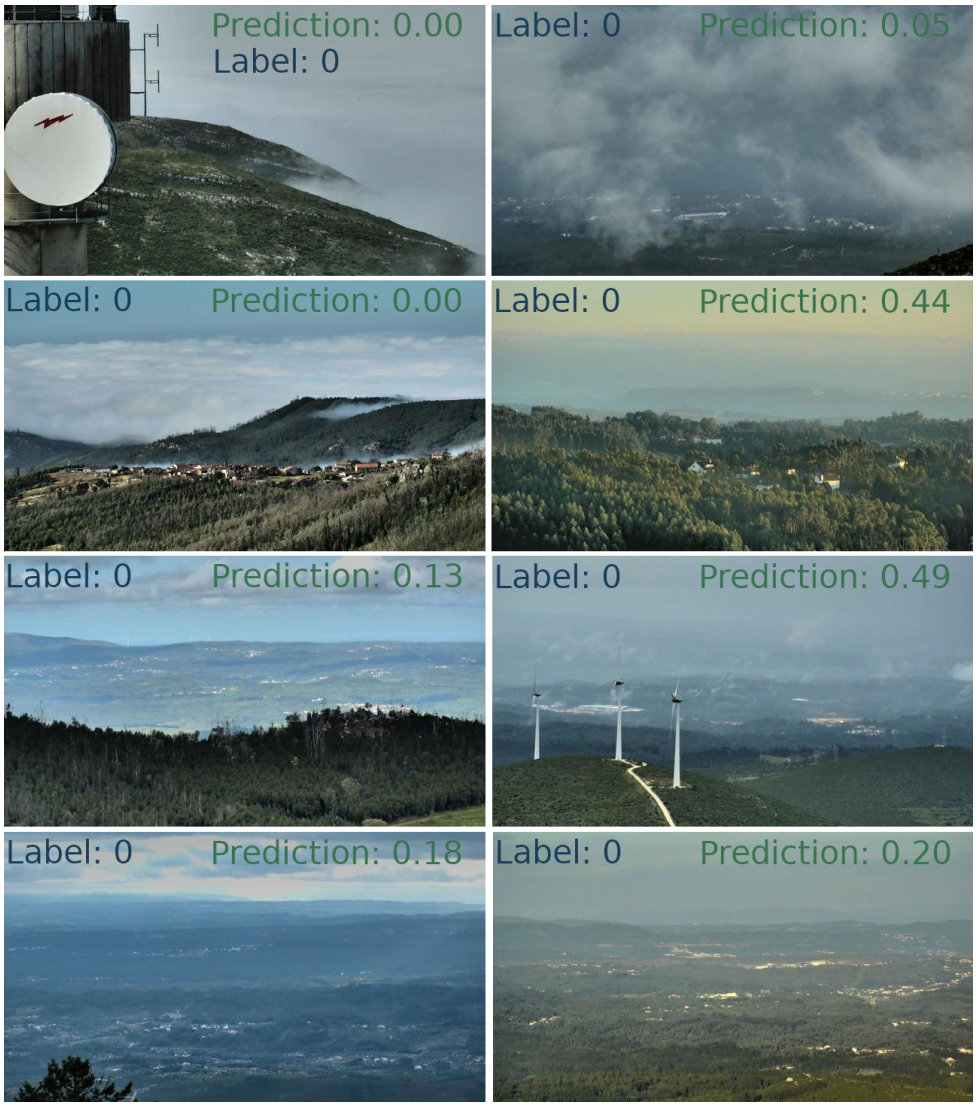

Figure 5.7: Examples of True Negatives.

Table 5.7: Results comparison with other related works.

Method	Technique	Accuracy	Image type	Dataset
Jiao *et al.* 2019 [7]	YOLOv3	83%	Large size fires	Private
Frizzi *et al.* 2016 [8]	CNN	97.9%	Large size fires and smoke areas	Private
Yin *et al.* 2017 [9]	CNN	98.1%	Large smoke areas	Private
Wang *et al.* 2020 [32]	Computer vision	>99%	Small smoke columns	Private
Muhammad *et al.* 2018 [30]	AlexNet	94.4%	Fire and smoke frames	Mivia [54]
Foggia *et al.* 2015 [55]	Computer vision	93.6%	Fire and smoke frames	Mivia [54]
Lascio *et al.* 2014 [56]	Computer vision	92.6%	Fire and smoke frames	Mivia [54]
Proposed	**EfficientNet**	**98.2%**	**Small smoke columns**	**Private**

with fire images. Then, they tested the model using a UAV to perform fire detection. Using 60 test images, the detection rate achieved 83%. Despite the model being able to pinpoint smoke and fire

locations using bounding boxes, the dataset consisted in large size fires, whereas this work focuses on the early detection of small smoke columns. In Frizzi *et al.* 2016 [8], the authors use CNNs to perform smoke detection in images. The classification accuracy on the test dataset is 97.9%, which is similar to the one achieved in our experiments (98.2%). However, the images shown in the respective paper also consist in large fires with big flames and smoke columns. Yin *et al.* 2017 [9] also uses CNNs for smoke detection. The examples from the smoke training dataset represent images where only smoke is visible, rather than images with small smoke areas like the ones used in this thesis. The accuracy rate presented is 98.1% using a total of 1505 images for testing purposes. Wang *et al.* 2020 [32] aims to detect early-stage fires, making use of several computer vision techniques such as moving region detection, smoke-colour analysis and sharp edge detection to identify small smoke plumes. The results presented show a $>$99% accuracy rate. Despite the promising results, the datasets that were used in this work were not disclosed. It is also important to mention that the number of smoke examples represents only about 2% of the total number of images in each of the three datasets, which sets the baseline for a naive classifier at 98% accuracy when he classifies all images as non-smoke.

There is also a public fire detection dataset, the Mivia dataset [54], that is composed of 14 videos that include fire and smoke frames captured indoor and outdoor and also 17 videos with no events of interest. This dataset includes some images with small smoke areas that are similar to the ones used in this work. In Muhammad *et al.* 2018 [30], the authors use the AlexNet architecture pre-trained in the ImageNet dataset to detect fire and smoke in images. The network is fine-tuned using a target dataset and the Mivia dataset was used to evaluate the proposed approach. The accuracy presented is 94.4% after fine-tuning the network and 90.1% without the fine-tuning phase. Foggia *et al.* 2015 [55] uses several methods to identify the presence of fire: colour-based, shape analysis and movement analysis in consecutive frames. Then, these features are combined to produce a single output. To assess the quality of the proposed approach, the authors achieved 93.6% accuracy in the Mivia dataset. In Lascio *et al.* 2014 [56], the pixels corresponding to moving objects are extracted from the images by using a detection algorithm. Then, two different kinds of information, respectively based on colour (Colour Threshold) and movement (Connected Component Filter and Disorder Evaluation), are properly combined by a multi-expert system to identify the presence of fire in videos. To test the approach, once again the Mivia dataset was used, and the results showed a 92.6% accuracy.

In summary, despite most of the works being focused on detecting large areas of smoke or flames and the datasets used were not public, the proposed approach used in this dissertation achieved a 98.2% accuracy whilst, simultaneously, being targeted to detect small emerging fires.

6

Conclusion

Contents

6.1 Contributions 60
6.2 Future Work 60

This chapter starts by listing the main conclusions and contributions of this dissertation (section 6.1) and then presents some approaches that could be explored in future works (section 6.2).

6.1 Contributions

This MSc thesis presents a deep learning method that allows the early detection of forest fires, commonly represented as small smoke areas, using images taken from multiple surveillance cameras. This work is divided into three main stages: (i) data pre-processing, (ii) image generation and (iii) image classification.

The data pre-processing stage includes the several steps performed to create the datasets used to train and test the models in both image generation and image classification. The data provided in the INOV dataset was used to locate and calculate the smoke area in the fire images. Regarding image classification, the image subdivision in several blocks allowed the study of this approach when compared to performing classification in regular images. When it comes to image generation, the smoke area location in the fire images was important to extract the smoke plumes that were used to train the image generation network.

The image generation phase was performed in the StyleGAN2-ADA architecture using two methods. In the first method, the goal was to use entire smoke images to train the network and try to generate fake smoke images that resemble real ones. This approach was not very successful, as the majority of the fake images generated either did not include any smoke areas or contained blurry areas. Since the results achieved did not allow to include these images in real datasets, a new method was tested, where the objective was to generate only small smoke plumes that later could be merged with regular images to create artificial fire images. This method achieved a similar FID score (12.61) compared to the baseline scores reached by the original StyleGAN2-ADA authors.

The proposed approach in the image classification stage compared the results obtained using the Xception, InceptionResNetV2 and EfficientNet architectures pre-trained in the ImageNet dataset. These models were then fine-tuned using two different transfer learning methods and several target datasets. The dataset used to test the quality of these models included mainly images with small size smoke areas, to assess the performance when detecting early-stage fires. The best result was obtained using the EfficientNet-B6 network and achieved a 98.2% accuracy.

6.2 Future Work

There are a few approaches that can be explored in future works. For instance, this work was performed using regular fire and non-fire images that were sampled from videos captured by the surveillance cameras. An interesting approach is to test and make the necessary adjustments to apply the image classi-

fication process in live real-world scenarios, for example in the CICLOPE project [3], taking into account processing performance and other real-life constraints. Regarding image generation, the generated images in this work were not used in the training datasets during image classification and, therefore, there are also a few approaches that can be taken into account. The first approach is to explore new ways to generate viable smoke images that can be used in real datasets. The second approach could be exploring methods that can be used to merge two different input images, to create a realistic output image, such as by using the SinGAN network [57]. In this case, the goal is to merge small smoke plumes with normal images, to generate artificial smoke images. Regarding the image classification process, there are a couple of scenarios that were not taken into account in this work, due to the lack of examples in the original dataset. For instance, training the model to detect fires at night or in other adverse conditions could be an interesting approach. One other way that has been recently used to monitor wildfires, to complement the use of surveillance cameras, is by using UAVs. It would also be interesting to test the model using aerial images captured from this source and make the necessary adjustments to allow the UAV to automatically detect the presence of a fire.

Bibliography

[1] PORDATA. (2020, May) Forest fires and area burnt. Last Accessed: 17-10-2021. [Online]. Available: https://www.pordata.pt/en/Europe/Forest+fires+and+area+burnt-1374

[2] European Environment Agency. (2021, June) Forest fires in Europe. Last Accessed: 17-10-2021. [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/forest-fire-danger-4/assessment

[3] CICLOPE. Ciclope: Homepage. Last Accessed: 17-10-2021. [Online]. Available: https://ciclope.com.pt/

[4] N. True, "Computer vision based fire detection," 2009.

[5] P. V. K. Borges and E. Izquierdo, "A probabilistic approach for vision-based fire detection in videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, pp. 721–731, 2010.

[6] X. Qi and J. Ebert, "A computer vision-based method for fire detection in color videos," *International journal of imaging and robotics*, vol. 2, pp. 22–34, 2009.

[7] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu, and D. Liu, "A deep learning based forest fire detection approach using uav and yolov3," in *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, 2019, pp. 1–5.

[8] S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 877–882.

[9] Z. Yin, B. Wan, F. Yuan, X. Xia, and J. Shi, "A deep normalization and convolutional neural network for image smoke detection," *IEEE Access*, vol. 5, pp. 18 429–18 438, 2017.

[10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[11] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *AAAI*, 2017.

[12] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *ArXiv*, vol. abs/1905.11946, 2019.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[14] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” 2020.

[15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2015.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” *ArXiv*, vol. abs/2010.11929, 2021.

[21] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[22] A. Bochkovskiy, C.-Y. Wang, and H. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *ArXiv*, vol. abs/2004.10934, 2020.

[23] Pierrick Rugery. (2020, September) Explanation of YOLO V4 a one stage detector. Last Accessed: 17-10-2021. [Online]. Available: https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[26] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019.

[27] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *ArXiv*, vol. abs/1802.05957, 2018.

[28] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *ArXiv*, vol. abs/1809.11096, 2019.

[29] G. Lin, Y. Zhang, G. Xu, and Q. Zhang, "Smoke detection on video sequences using 3d convolutional neural networks," *Fire Technology*, pp. 1–21, 2019.

[30] K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," *Neurocomputing*, vol. 288, pp. 30–42, 2018.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.

[32] A. Genovese, R. D. Labati, V. Piuri, and F. Scotti, "Wildfire smoke detection using computational intelligence techniques," in *2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings*, 2011, pp. 1–6.

[33] R. Donida Labati, A. Genovese, V. Piuri, and F. Scotti, "Wildfire smoke detection using computational intelligence techniques enhanced with synthetic smoke plume generation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 1003–1012, 2013.

[34] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[35] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *ArXiv*, vol. abs/1508.06576, 2015.

[36] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

[37] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in *NIPS*, 2016.

[38] G. R. Koch, “Siamese neural networks for one-shot image recognition,” 2015.

[39] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *ICML ’07*, 2007.

[40] K. Wang, X. Gao, Y. Zhao, X. Li, D. Dou, and C. Xu, “Pay attention to features, transfer learn faster cnns,” in *ICLR*, 2020.

[41] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, “Scaling vision transformers,” *ArXiv*, vol. abs/2106.04560, 2021.

[42] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *ICML*, 2014.

[43] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *ArXiv*, vol. abs/1411.1792, 2014.

[44] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[45] Chollet, Francois. (2020, April) Keras: Transfer learning & fine-tuning. Last Accessed: 27-10-2021. [Online]. Available: https://keras.io/guides/transfer_learning/

[46] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116, 2020.

[47] INOV. CICLOPE: Sistema integrado de videovigilância de grandes áreas. Last Accessed: 17-10-2021. [Online]. Available: https://www.inov.pt/project/ciclope/index.html

[48] Google. Google Cloud Platform. Last Accessed: 17-10-2021. [Online]. Available: https://cloud.google.com/

[49] NVlabs. (2021, February) StyleGAN2 with adaptive discriminator augmentation (ADA) — Official TensorFlow implementation. Last Accessed: 17-10-2021. [Online]. Available: https://github.com/NVlabs/stylegan2-ada

[50] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *NIPS*, 2017.

[51] Nvlabs. (2021, September) Flickr-Faces-HQ Dataset (FFHQ). Last Accessed: 17-10-2021. [Online]. Available: https://github.com/NVlabs/ffhq-dataset

[52] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop,” 2016.

[53] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “Stargan v2: Diverse image synthesis for multiple domains,” 2020.

[54] Mivia. Mivia Fire Detection Dataset. Last Accessed: 20-10-2021. [Online]. Available: http://mivia.unisa.it/datasets/video-analysis-datasets/fire-detection-dataset

[55] P. Foggia, A. Saggese, and M. Vento, “Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, pp. 1545–1556, 2015.

[56] R. D. Lascio, A. Greco, A. Saggese, and M. Vento, “Improving fire detection reliability by a combination of videoanalytics,” in *ICIAR*, 2014.

[57] T. R. Shaham, T. Dekel, and T. Michaeli, “Singan: Learning a generative model from a single natural image,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4569–4579, 2019.