Technical University of Crete
School of Electrical and Computer Engineering

DIPLOMA THESIS

Wildforest Fire Detection using Deep Learning and Fusion Techniques on Aerial Image Datasets

EMMANOUIL ZACHARIADIS

Committee

Professor Michalis Zervakis (Supervisor)

Professor Michail G. Lagoudakis

Associate Professor Apostolos Voulgarakis (School of CEE)

CHANIA, JULY 2023

Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Εντοπισμός πυρκαγιάς σε δασικές περιοχές, χρησιμοποιώντας τεχνικές βαθιάς μάθησης και ενοποίησης δεδομένων σε εναέριες εικόνες

Εμμανουήλ Ζαχαριάδης

Επιτροπή

Καθηγητής Μιχάλης Ζερβάκης (Επιβλέπων)

Καθηγητής Μιχαήλ Γ. Λαγουδάκης

Αναπληρωτής Καθηγητής Απόστολος Βουλγαράκης (Τμήμα ΧΗΜΗΠΕΡ)

Χανιά, Ιούλιος 2023

Acknowledgments

At this point, I would like to thank Prof. Michalis Zervakis and Dr. Marios Antonakakis who stood by my side during this long journey and wisely guided me to paths that I did not know I wanted to follow.

In addition, I would like to thank my family who have been my biggest supporters throughout this difficult, but rewarding, period of my studies at the Technical University of Crete.
Thank you, Niko, Kiriaki and Zack.

Abstract

Greece, like other countries, suffers every year from intense fires which inevitably cause losses. In the best-case scenario, these losses are materialistic only, but there are many cases where the consequences are not limited to those losses. This kind of destruction has a crucial impact on firefighters and civilian lives, animal extinction, and forest degradation. Considering scientists' predictions that fires will increase every year due to climate change, many prevention systems have been set up to avoid this kind of situation. This thesis focuses on developing a detection approach that can classify images of forests in “Fire” and “non-Fire” cases with input from both RGB and Infrared (IR) cameras. Along with this work, an architecture is proposed that increases the possibility to prevent these previously mentioned consequences. The new deep learning architecture is called *ShRe-Xception* (Short Recursive), inspired by already existing Xception network architectures (i.e., small-arch Xception and original Xception Network). In this study, an experimental process takes place around how to train a neural network with two different datasets and the relevance of its architecture. First, transfer learning is performed on the small-arch Xception network with input from either RGB or IR images, and then, the same structure is trained with all the RGB images from the very beginning. Also, the proposed ShRe-Xception network is trained by using all RGB images from the very beginning and then retrained with infrared frames. Datasets used for training and testing purposes contain “Fire” and “non-Fire” images of forests that are captured from UAVs and uploaded to the IEEE portal. When testing the above models, the evaluation accuracy was RGB: 90.10% and IR: 99.31% after initial training and ShRe-Xception model, RGB: 77.13% and IR: 94.47% after transfer learning on small-arch Xception, and RGB: 84.86% and IR: 29.19% when initially training the small-arch Xception. The above results are described and analyzed within the present thesis, as many other operations were performed in terms of experimentation and results. In general, the latter model came to be the most accurate one for images of both spectrums in our experiments. The ShRe-Xception can potentially play a vital and efficient role in real-time fire detection during aerial surveillance of wild forests.

Περίληψη

Είναι γεγονός ότι η χώρα μας, όπως και άλλες χώρες, υποφέρει κάθε χρόνο από έντονες πυρκαγιές που αναπόφευκτα προκαλούν μεγάλες απώλειες. Στην καλύτερη περίπτωση, οι απώλειες αυτές είναι μόνο υλικές, αλλά υπάρχουν πολλές περιπτώσεις όπου οι συνέπειες δεν περιορίζονται μόνο σε αυτό το φάσμα. Σε πολλές περιπτώσεις, οι καταστροφές αυτές έχουν κρίσιμες επιπτώσεις στις ζωές των πυροσβεστών και των πολιτών, στην εξαφάνιση των ζώων και στην καταστροφή των δασών. Λαμβάνοντας υπόψη τις προβλέψεις των επιστημόνων ότι οι πυρκαγιές θα αυξάνονται κάθε χρόνο λόγω της κλιματικής αλλαγής, έχουν δημιουργηθεί πολλά συστήματα πρόληψης για την αποφυγή τέτοιου είδους καταστάσεων. Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη ενός συστήματος ανίχνευσης που μπορεί να διαχωρίσει εικόνες δασών σε περιπτώσεις "πυρκαγιάς" και "μη πυρκαγιάς" με είσοδο τόσο από κάμερες RGB, όσο και από κάμερες υπερύθρων (IR), αποτρέποντας δυνητικά εκτεταμένες καταστροφές. Μέσα στην εργασία μας περιέχεται και ο σχεδιασμός ενός νέου μοντέλου παραγμένου από την ομάδα μας, ο οποίος πιστεύουμε ότι αυξάνει τις πιθανότητες για την αποφυγή μιας δασικής καταστροφής. Ο σχεδιασμός μας ονομάζεται ShRe-Xception (Short Recursive) και είναι ένα συνονθύλευμα που χρησιμοποιεί χαρακτηριστικά, τόσο από μια υπάρχουσα μικρή - αρχιτεκτονική του δικτύου Xception (small-arch Xception), όσο και από την αρχική έκδοση του δικτύου Xception. Αυτή η μελέτη περιλαμβάνει μια πειραματική διαδικασία γύρω από τον τρόπο εκπαίδευσης ενός νευρωνικού δικτύου με δύο διαφορετικά σύνολα δεδομένων και τον ρόλο που έρχεται να παίξει η αρχιτεκτονική του στην μετέπειτα απόδοση. Αρχικά, εκτελούμε transfer learning στο small-arch Xception με είσοδο, είτε από εικόνες RGB, είτε από εικόνες IR και στη συνέχεια, παίρνουμε την ίδια αρχιτεκτονική και την εκπαιδεύουμε με όλες τις εικόνες RGB από την αρχή, εκτελώντας αργότερα την επανεκπαίδευση με θερμικές εικόνες. Εκπαιδεύουμε επίσης το προτεινόμενο δίκτυο ShRe-Xception, χρησιμοποιώντας όλες τις εικόνες RGB από την αρχή και επανεκπαιδεύοντάς το με θερμικές εικόνες στη συνέχεια. Τα σύνολα δεδομένων που χρησιμοποιούνται για σκοπούς εκπαίδευσης και δοκιμής περιέχουν εικόνες "πυρκαγιάς" και "μη πυρκαγιάς" από δάση που έχουν ληφθεί από UAV και έχουν αναρτηθεί στην IEEE dataport. Κατά τη δοκιμή των παραπάνω μοντέλων, η ακρίβεια των δοκιμών ήταν RGB: 77,13% / IR: 94,47% μετά το transfer learning στο small-arch Xception, RGB: 84,86% / IR: 29,19% κατά την αρχική εκπαίδευση του small-arch Xception και RGB: 90,10% / IR: 99,31% μετά την αρχική εκπαίδευση στο μοντέλο ShRe-Xception. Τα παραπάνω αποτελέσματα είναι ενδεικτικά, καθώς υλοποιήθηκαν και πολλές άλλες τεχνικές. Σε γενικές γραμμές, το τελευταίο μοντέλο αποδείχθηκε το πιο ακριβές για εικόνες και των δύο φασμάτων στα πειράματά μας. Το ShRe-Xception μπορεί ενδεχομένως να διαδραματίσει ζωτικό και αποτελεσματικό ρόλο στην ανίχνευση πυρκαγιάς σε πραγματικό χρόνο κατά την εναέρια επιτήρηση των δασών.

TABLE OF CONTENTS

Table of Figures

Table of tables

Chapter 1 Introduction

Wildforest fires are a frequent and major problem that occurs every year during summer, spring, and autumn periods, causing severe damage to forests, animal wildlife, residential areas, and even people's lives [1]. This kind of destruction is currently difficult to be detected at an early stage, as the numbers indicate, creating an inconvenient situation for the competent authorities to manage. To accomplish the forest fire early detection task, many techniques have been introduced up to this time, such as using human observatories, Unmanned Aerial Vehicles (UAV), or Unmanned Ground Vehicles (UGV) and installing ground sensors. This work is focused on the fire detection problem and this is done by using convolutional neural networks (CNNs) and images acquired from sensors of different types (RGB or IR) in order to produce a model that classifies images to "Fire" and "non-Fire" cases. In this work, aerial images are exclusively used, as UAVs offer a wider Field of View (FOV) and easier obstacle avoidance. Focused on detecting forest fires from above, each dataset used for the development of the model is gathered from a UAV with a panoramic viewing angle.

1. 1 *Unmanned Aerial Vehicles (UAVs)*

An Unmanned Aerial Vehicle is an aircraft commonly known as a drone, without any kind of crew on board. It can be remotely operated from a ground station or follow an entirely automated route that is predefined by the pilot/operator [3]. The first produced models were used mainly in military operations but in recent years there has been a wide use in aerial photography, forest fire monitoring, agriculture, and surveillance. In our case, forest fire monitoring, UAVs can be used in early detection, containment, and the extinguishing of any

Figure 1 Flying UAV on duty [2].

fire case. Their high manoeuvrability as well as the easy take-off and landing can be very useful characteristics in terrains where it is considered dangerous or impossible for firefighters to reach [4].

The use of UAVs in all the above situations became so popular due to the great evolution that has been made in the camera industry. As for the RGB optical sensors, there has been a huge

advancement in image resolution making it easier for both neural networks to perform object detection or pattern recognition processes and for the human eye to catch any unexpected event.

As for thermal imaging, both the growth in camera quality and the reduction of its own cost made it more convenient for drone companies to enhance them in their products. Recently, a dual camera (named FLIR Hadron™ Integrated RGB/Thermal Module) has also reached the production process offering images from two spectrums in one common module. However, time is still needed to further develop such technologies and the choice of individual cameras for each drone is the dominant one, leaving the question of which one is best for each company to use.

1. 2 *RGB vs IR spectrum*

The majority of previous studies focus on the RGB spectrum, as RGB cameras have some useful advantages. Firstly, it is a matter of price. Color cameras appear to have significantly lower prices than thermal ones making them more accessible to manufacturing companies. Additionally, when in daylight, these sensors can produce high-resolution frames that can be used in machine vision systems [5], [6]. Moreover, as the RGB images have more resolution, many other attributes of the image can be obtained such as texture and luminosity, making it easier for a neural network to perform pattern recognition. On the other hand, when a UAV is patrolling in a dark environment, like a forest or a mountain that cannot be artificially illuminated, the RGB sensor has nothing to offer to our system [7]. Hence, thermal cameras should be used in order to prevent fires that break out at night. These sensors, except for night vision, can provide high performance in many different weather conditions (such as rain, cloudy weather, and smoke). More specifically, when monitoring a burning forest that fills with smoke the entire image, the IR camera can see below this layer and detect the fire's position, offering the maximum help to firefighters [8]. However, it should be noted that these sensors do not have high discriminating capabilities as described in "The Johnson Criterion" for thermal images [9]. To summarize, as both the above sensors have many advantages to offer to our system, there will be a need to use images of either spectrum. These images will be used to train and test different neural networks to extract the best possible solution (model architecture) for our system as we will see in the sections below.

1. 3 *Convolutional Neural Networks*

The image classification process would not be possible if it were not for neural networks. Convolutional Neural Networks (CNNs) are one of many different deep learning algorithms that are applied to fields such as object detection and pattern recognition. They provide features such as speed processing and the capability to learn by analysing a set of inputs [10]. In this study, the fire detection problem is mainly explored with the use of a specific neural network architecture named Xception network [11], presented in Fig. 2. More precisely, a recent study [1] introduced the use of a smaller architecture of Xception network (small-arch Xception) to solve the same task. This architecture is presented in Fig. 3 and has been scaled down, in comparison to the original version, to a more condensed design to fit in embedded systems with limited capabilities such as Nvidia's Jetson series. In our work, many experiments were conducted using this previously designed small-arch Xception network by using the same datasets, adding extra sets of RGB and IR images to examine its robustness, and finally transforming it into a new network, produced by our team, that is later to be discussed.

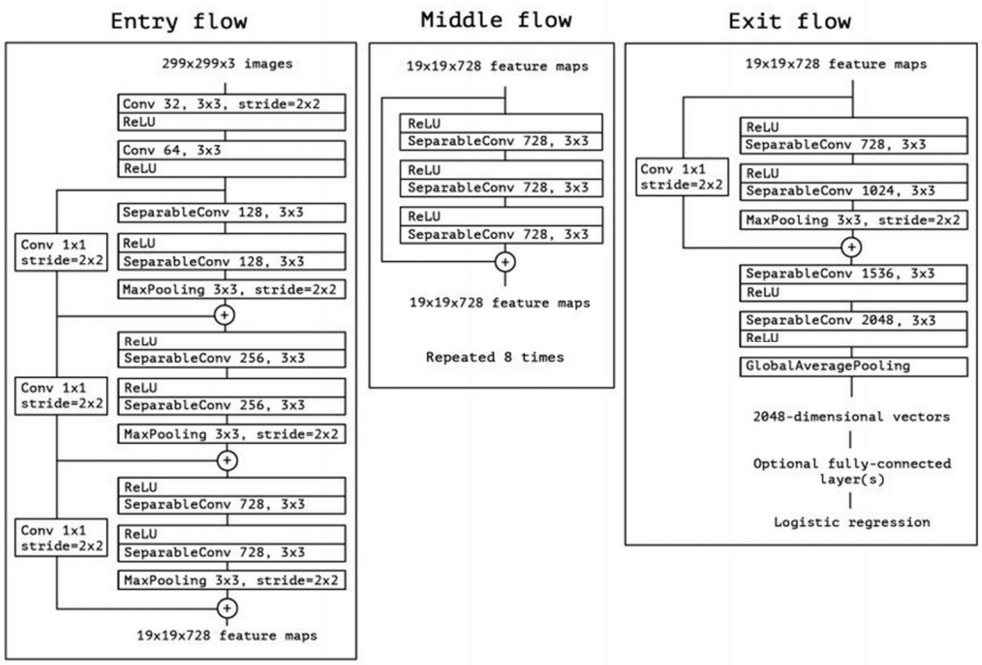


Figure 2 Xception network original architecture.

1. 4 *Background work*

As mentioned before, some studies have taken place before ours that deal with the same problem. However, as we aim for forest patrolling with drones, the images used and the architecture to choose will be based on this particular situation. This previous study [1] uses a dataset to train and test the small-arch Xception that has mainly this characteristic of images and goes by the name of FLAME dataset [12]. The authors worked only on a RGB set of images and the reported classification testing results were of low performance (i.e., accuracy: 76.23%). Additionally, some other recent studies are having noticeable results.

Still, the majority of images' background includes urban regions and not forest areas. These works are mainly concentrated on the use of RGB datasets. Apart from these, in works such as [13] where authors test different fire detecting algorithms along with proposing their model or [14] where authors experiment with different already existing neural networks, images are captured from cameras placed on ground stations. Even in [15] the authors make use of aerial images, the testing dataset is of limited size. Furthermore, there is not any reference to thermal imaging. In contrast to ours, where two very big datasets are used and many experiments with infrared classification are conducted, only the accuracy is a matter of comparison, and no other metric was presented. Additionally, it should be mentioned that these works use high-end hardware to produce these sky-scraping results, making it impossible to use these models in small embedded systems that can be carried in a drone's compartment. Thus, it was necessary to introduce a system that combines UAV-captured images, RGB and IR spectrum, high accuracy, and a fast robust operation on machines with lower technical specifications.

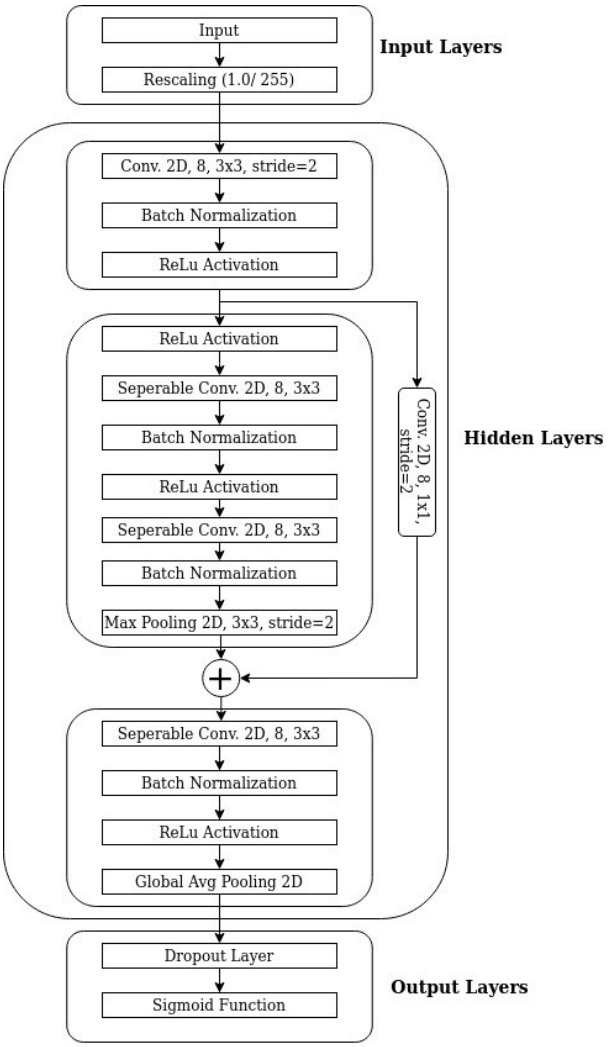


Figure 3 small-arch Xception architecture.

1. 5 *Novelty of the present thesis*

In the proposed work, the small-arch Xception published in [1] is utilized to build an optimized network with potentially higher performance. We aim to classify RGB or IR images of "Fire" and "non-Fire" cases using forestry-based images acquired from a drone flying at higher altitudes. Through an experimenting process where small-arch Xception's architecture was changed and differentiated, we end to propose a new structure of the Xception network, the ShRe-Xception (Short Recursive) which is a condensed version inspired by the original Xception network [11] and the small-arch Xception network proposed by Google Keras [16]. This version on the one hand has been compressed to work on systems with limited capabilities and on the other side it is, extended, to increase its performance compared to the last model. The design of the proposed architecture is presented below in Figure 4.

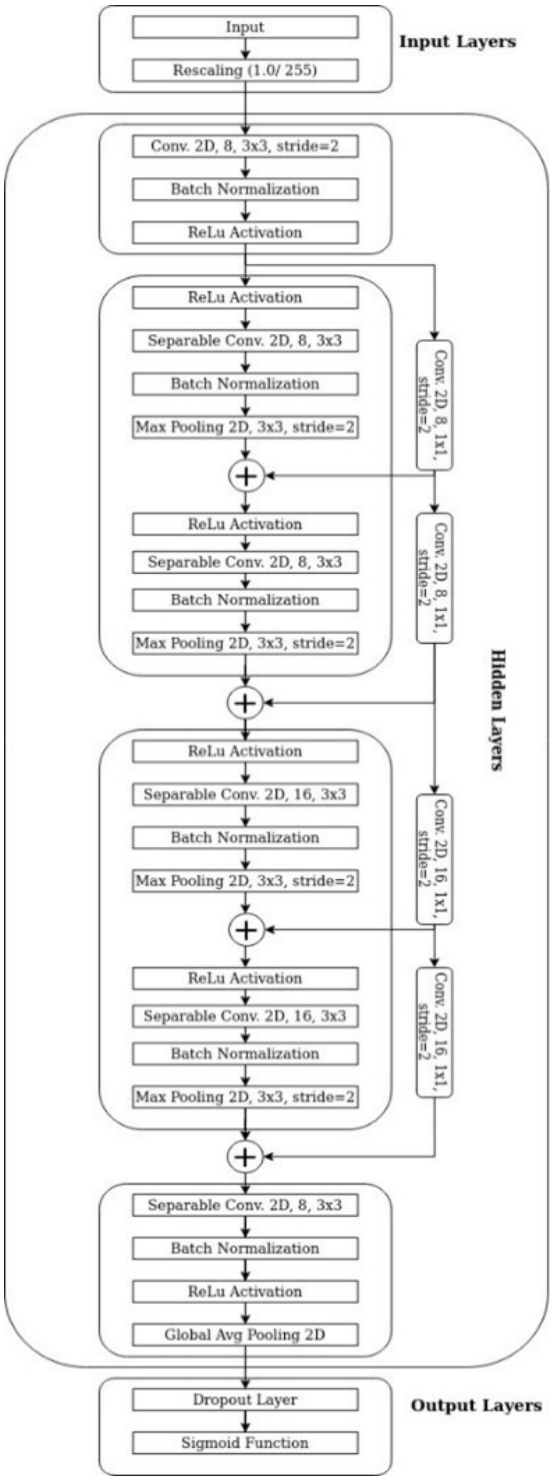


Figure 4 ShRe-Xception architecture.

Chapter 2 Theoretical Background

2. 1 *Artificial Intelligence*

Artificial intelligence (AI) has emerged as a pioneering field of research and technological development that came to improve many aspects of human life. With the recent advancements in computing power, machine learning algorithms, and data availability, AI has become a driving force behind transformative innovations across industries. It consists of any technique that enables a machine to process data to drive some future decision, mimicking human behaviour.

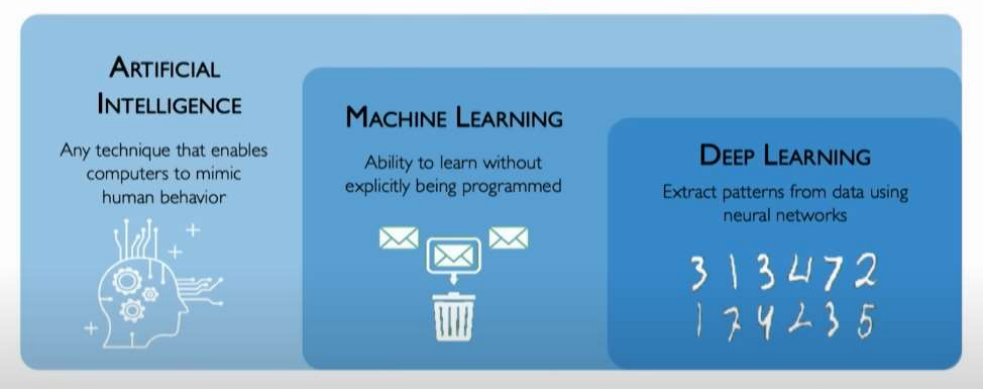


Figure 5 From MIT Introduction to Deep Learning | 6.S191

2. 2 *Machine learning*

Machine Learning (ML) is a subset of AI and describes the ability of a machine to learn through experiments. More specifically, by providing the questions and the answers to a problem, the system can automatically create its own rules in decision-making, without being explicitly programmed. This approach allows machines to automatically analyse vast amounts of data, detect patterns, and make accurate predictions. A subset of ML is Deep Learning (DL)

2. 3 *Deep Learning*

As mentioned before Deep learning is a field of machine learning, that offers noticeable advancements in modeling and detecting complex patterns in data. Deep learning architectures, analyze data with a logical structure inspired by the structure and function of the human brain and are characterized by multiple layers of interconnected artificial neurons. The training process can be done both through supervised and unsupervised learning. To achieve these Deep Learning applications, a layered structure of neurons called Convolution Neural Networks (CNNs) is used, leading to a process of learning that's far more capable than that of standard machine learning models. To make this possible there are two main things that someone needs to train and test a DL model, data availability and computing power. There is a need for a great amount of data to train a network since CNNs learn by example. The network must be passed by many different examples to start recognizing objects and many more in order to become robust enough to set the right rules for its decision-making. Secondly, DL needs substantial

computing power. Yet, with the emergence of cloud computing infrastructure and high-performance GPUs, the time for training a network is significantly reduced [17].

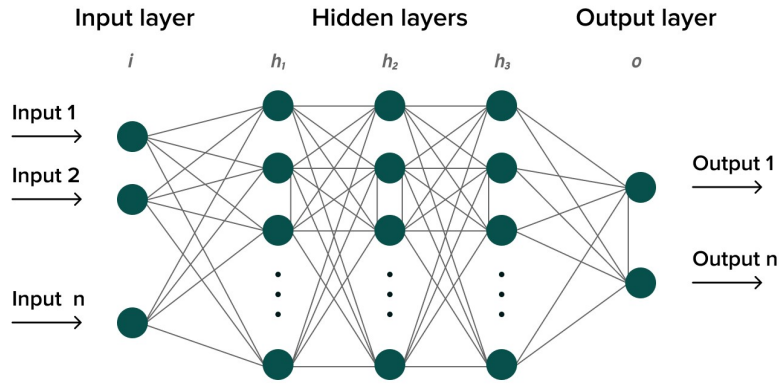


Figure 6 Convolution Neural Network representation [18].

2. 4 *Convolution Neural Networks*

Convolution Neural Networks (CNNs) are one of the Deep Learning algorithms designed for processing and analysing structured grid-like data, such as images. CNNs have gained significant popularity and achieved state-of-the-art performance in various computer vision tasks, including image classification, object detection, and image segmentation. One of the key features of CNNs is their ability to automatically learn spatial hierarchies of features from raw input data, enabling them to capture local patterns, global structures, and semantic representations. CNNs are composed of several layers, such as convolutional layers, pooling layers, and fully connected layers. In the convolutional layers, kernels are convolved with the input data to extract the most significant features and create feature maps. The pooling layers help reduce the spatial dimensions of the feature maps, enabling the network to be more robust to translations and variations. Finally, the fully connected layers combine the extracted features and perform the classification.

In our work, the input tensor for the network to classify is an image of 254px254p resolution. This resolution is a little smaller than expected nowadays but it is due to the multitude of datasets that we were able to find, to train and test our networks. As for the color channels of the image, it can be one or three depending on the way that the images are captured.

2. 4. i *Convolution Layer*

Passing the input process, the convolution layer is usually the first layer of a CNN where an image is convolved with a filter of specific dimensions called kernel. Kernels are small filters

that are applied to image's data through a sliding window that convolves across its width and height computing the dot product between image's pixel values and kernel's values, producing a feature map for this specific filter. Kernel's dimensions are specified from the beginning of the process and its depth depends on the image's depth.

Having a 2d image and one filter that will be 2d, as mentioned before, their convolution will generate a matrix that will be a 2d array. So, the summary is that the number of image's channels will determine the depth of the kernel. Now, the convolved image that will be produced after applying the filter will may have different dimensions compared to the initial one. When a n × n image convolves with a m × m filter without using padding, the convolved image dimensions will be (n-m+1) × (n-m+1). Having a network that is quite deep, this process can lead to shrinking the image as many convolutions are performed iteratively. Additionally, as the image gets cropped step-by-step, its edges start to disappear and the neural network loses potentially relevant information, preventing the edge pixels to contribute as much to the feature detection. The padding process comes to solve this issue by extending the image sufficiently to maintain the original image dimensions after the convolution. So, 3 types of padding can be used:

1) Zero-padding
2) Symmetric padding
3) Replicate
4) Circular

1) Zero-padding: is a process where you put zeros everywhere around the image to extend it like Figure 7 below.

0	0	0	0	0	0	0	0	0	0
0	1	29	13	2	3	20	17	26	0
0	24	33	32	7	9	10	4	2	0
0	14	10	2	21	1	18	22	21	0
0	15	7	4	14	19	3	10	13	0
0	16	14	8	16	4	17	38	7	0
0	11	25	6	2	3	31	36	21	0
0	24	10	3	9	11	28	21	10	0
0	33	2	19	7	10	22	6	25	0
0	0	0	0	0	0	0	0	0	0

Figure 7 Zero padding example.

2) Symmetric padding: is the process where you mirror the edge pixels symmetrically to the extended part of the images as shown below in figure 8**.**

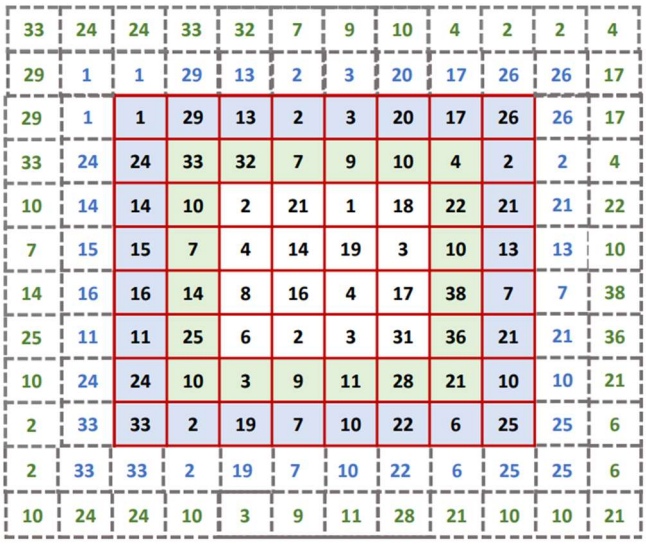


Figure 8 Symmetric padding example.

3) Replicate padding is the process where you grab the value of the edge pixel and use it to extend the image iteratively as presented in Fig 9.

Figure 9 Replicate padding example.

4) Circular padding: is the process where you grab the value of the edge pixel and use it to the symmetrically opposite position to extend the image iteratively as presented in Fig 10.

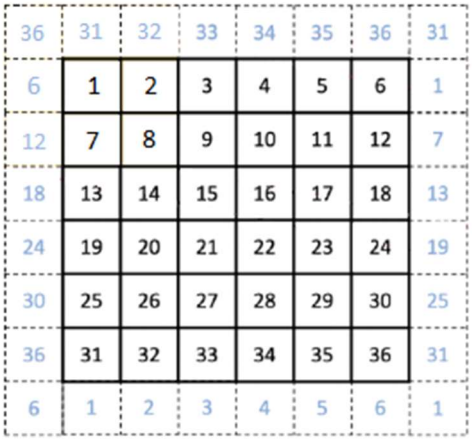


Figure 10 Circular padding example.

Moreover, there is another parameter that tends to affect the convolution process. It is called stride and it indicates the step (how many shifts) that the kernel should do after it is applied to a n x n window.

To conclude, a Convolution Layer has the 4 following main characteristics:

1) Kernel size
2) The number of image channels
3) Padding
4) Stride

It should be noted that in our work there are not only convolution layers used, but separable convolutional layers too. This type of layer is introduced in [6 paper], where the authors prove the similarity of those two previous layers but recommend the use of the second one since it is less cost-intensive in terms of complexity and parameter multitude.

2. 4. ii *Pooling Layer*

Pooling layers are one of the main components of Convolution Neural Networks. Their importance is based on the fact that they are responsible for the down-sampling procedure. Down-sampling is necessary as an image cropped or tilted that still contains important structural elements, without the fine detail, may not be successfully classified. Pooling layers consolidate the features learned by CNNs, making them invariant to translations. The objective is to progressively reduce the spatial dimension of the representation to minimize the network's parameters and computations.

Pooling involves selecting a pooling operation. The size of the pooling operation is almost always 2×2 pixels applied with a stride of 2 pixels. As a result, the pooling layer will necessarily divide the dimensions (Height, Width) by a factor of 2, thereby reducing the number of pixels available for the subsequent convolution operation.

Two common pooling functions are used:

1. Max or Maximum Pooling: is the function that calculates the maximum value for each block where the pooling window is applied. As Max Pooling retains the most prominent features of the feature map, the returned image is sharper than the original image.
2. Average Pooling: the function that calculates the average value of each block. In contrast to the previous category, Average Pooling retains the average values of features so, the returned image is smoother but with maintaining the main features of the image [19].

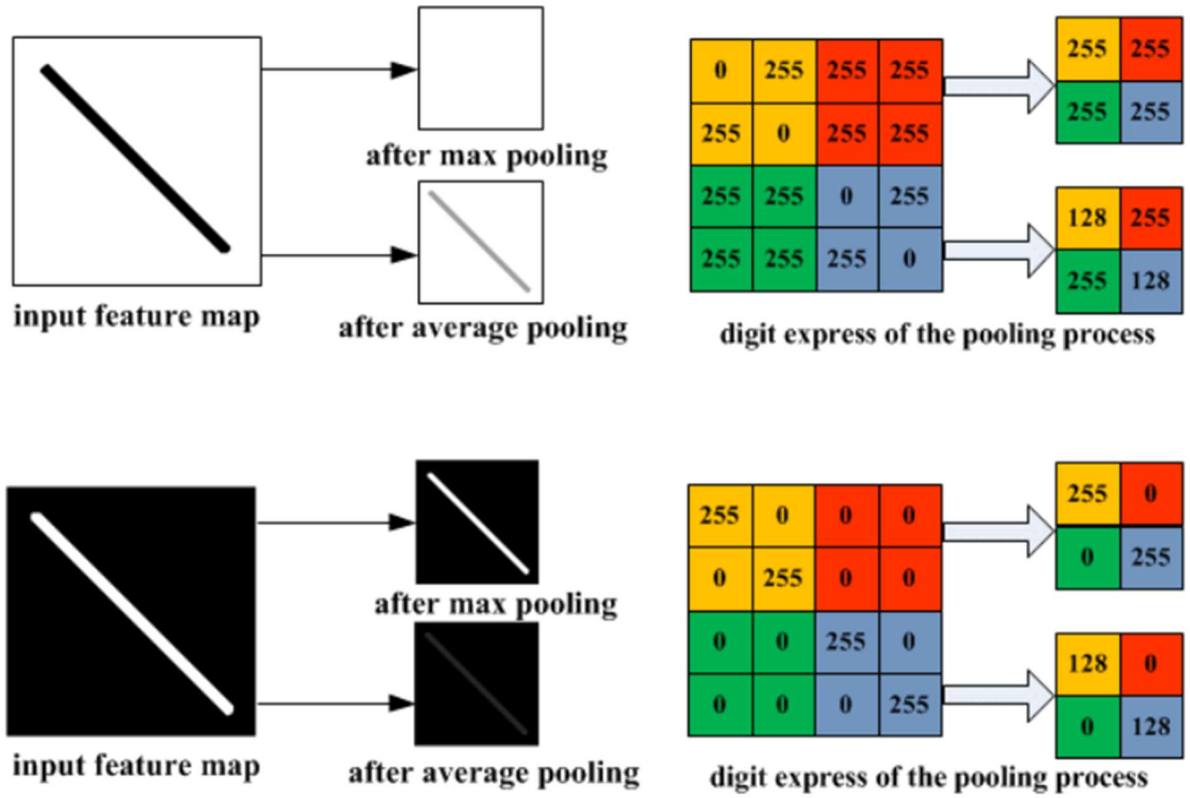


Figure 11 Illustrations of different pooling processes and their drawbacks [20].

As we can see from the previous image when we have a bright background the max pooling process loses the object to be detected (line) and transfers a blank image to the network. On the other hand, in a dark background, average pooling, could not maintain the bright line and additionally blurs it enough to start a disappearing process.

2. 4. iii *Fully Connected layer*

A Fully Connected layer (FC) is the last layer in the convolution neural network. This type of layer consists of two individual layers, one of which is an Affine Function and the other is a Non-Linear Function. The FC layer takes the output of the previous convolution or pooling layers, "flattens" them, and returns a single vector as an input to a Non-Linear function. It is important to note that the output after performing convolutions and after passing through pooling layers is a 3-d matrix. To flatten it, the output is passed through a global average pooling layer where each matrix value is stacked and the result is a single-dimension vector. The flattened vector is then passed to a fully connected layer which is expressed by:

$$y = g(wX + b)$$

where:

w is a vector containing the weights that are used.

X is the input vector.

b is the bias vector which is an additional set of weights in a neural network that corresponds to the output of a convolutional neural network when it has zero input.

g represents the activation function.

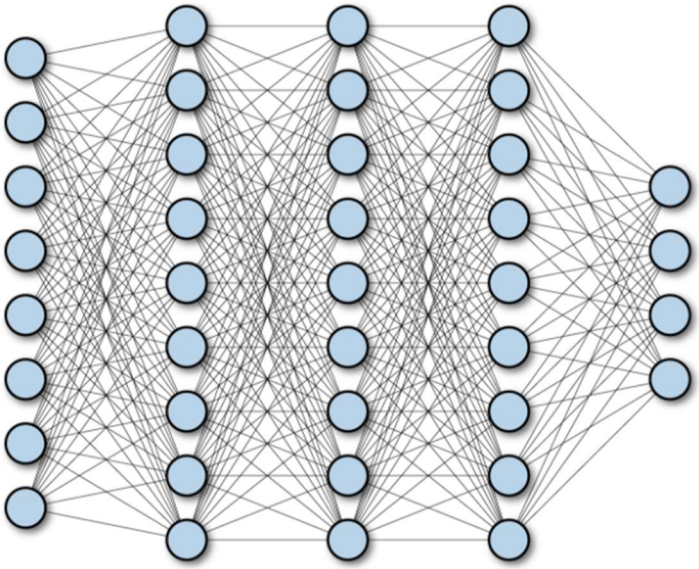


Figure 12 Fully Connected layer example [21].

This combination of layers ends with an Activation Function, as mentioned before. An Activation Function decides whether a neuron should be activated or not. This means that it decides whether the neuron's input is important or not in the process of prediction. There are two categories of Activation Functions, and they are listed as:

1. Linear Activation Functions.
2. Non-Linear Activation Functions.

Below are presented some examples of Activation Functions, to show how they work and the advantages and disadvantages behind their use in neural networks [22].

Binary Step Activation Function:

It is a linear function that relies on a threshold value. If the input value is greater than the threshold, the neuron is activated, or else is not activated, and the input value will not be transferred to the next layer.

It can be mathematically expressed as:

$$f(x) = f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

And graphically expressed as:

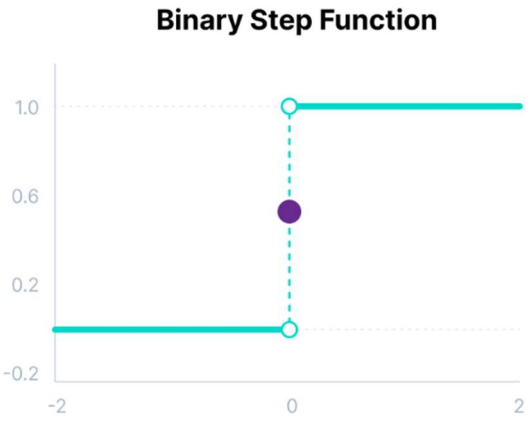


Figure 13 Binary Step Function

Linear Activation Function:

The Linear Activation Function is the function where the output is proportional to the input. It is also known as "no activation" or "identity function" as the input is multiplied by 1 and the function has nothing to do with the weighted sum of the input.

Mathematically it is expressed as:

$$f(x) = x$$

And graphically is expressed as:

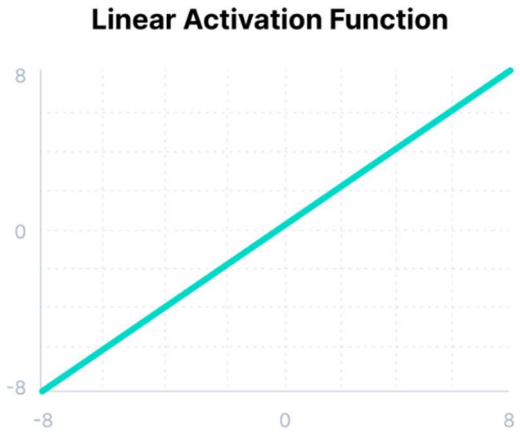


Figure 14 Linear Activation Function

Above, some linear activation functions were presented. Those functions are of limited power as they cannot create complex connections between the network's inputs and outputs. Non-Linear functions, on the other hand, have the power to revoke these limitations since by allowing the stack of multiple layers of neurons, the output can now have a non-linear

association with the input that is passed through multiple layers. Additionally, it allows backpropagation, which can offer a deeper understanding of which weights in the input neurons can provide a better prediction for our model. Driven by all the previously discussed advantages, some non-linear functions, which are essential to convolutional neural networks, are presented.

ReLU Activation Function:

ReLU (Rectified Linear Unit), while giving the impression of a linear function, has a derivative function that allows for backpropagation. Its main advantages are that it is computationally efficient and does not activate all the neurons at the same time. The neurons will be activated with linearity if the input is greater than zero, but not activated at all if the input is less than zero. So mathematically it is represented as:

$$f(x) = \max(0, x)$$

And its graphical representation is:

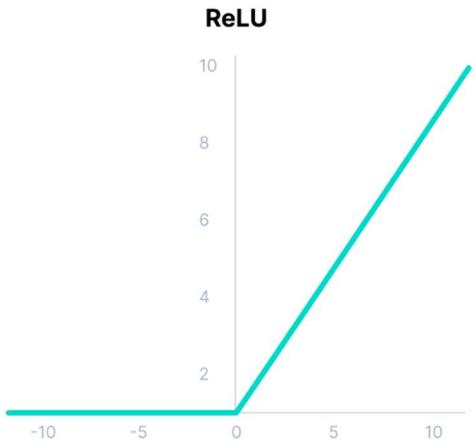


Figure 15 ReLU Activation Function

Unfortunately, the derivative when you browse the negative side of the graph is zero. This means that during backpropagation, some weights and biases of not activated neurons will not be updated and this can create neurons that are placed in the network but not ever activated (dead neurons). This decreases the model's ability to train from the data accurately. For this reason, some other versions of this activation function were created and will be presented below.

Leaky ReLU Activation Function:

The Leaky ReLU function is an optimized ReLU as it solves the neuron Dying problem as it has a small incline in the negative grid as can be seen in Fig 16. Its mathematical representation is very similar to the original version but with one difference in the negative inputs. More specifically the output is:

$$f(x) = \max(0.1 \cdot x, x)$$

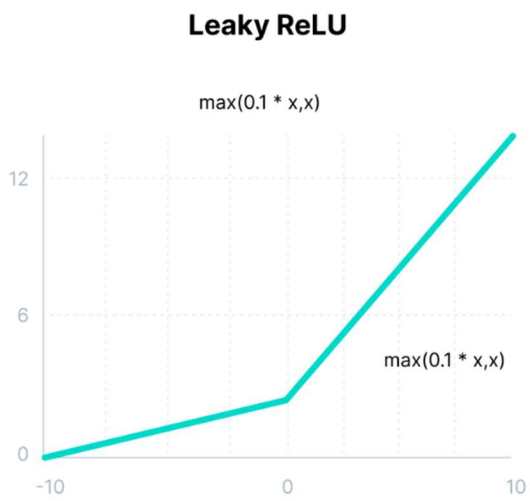


Figure 16 Leaky ReLU Activation Function

Parametric ReLU Activation Function:

The Parametric ReLU function is another version of the ReLu function that tries to solve the Dying problem. The difference between this version and the previous one (Leaky ReLU) is that it does not use a fixed factor for the incline as "0.1" was used before, but goes with the use of a variant incline (a*x), where "a" is a learned value that the network retrieves during backpropagation process. Its mathematical representation is:

$$f(x) = \max(a \cdot x, x)$$

And its graph is:

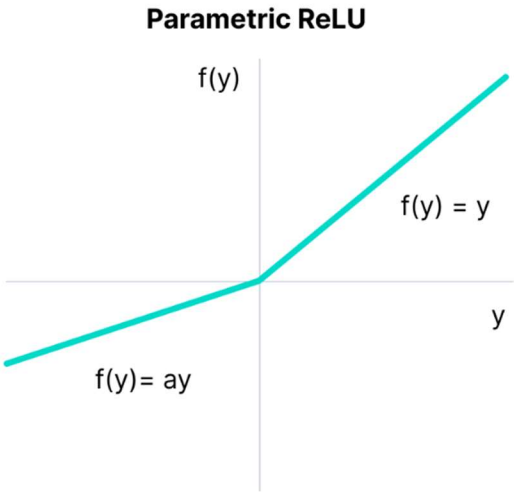


Figure 17 Parametric ReLU Activation Function

The Parametric ReLU function is used usually when Leaky ReLU fails to solve the Dying problem. However, we should bear in mind that this activation function will perform differently depending on the value of "a" which controls the incline of the line on the negative space.

Passing through different ReLU functions that should be used in the hidden layers, some other activation functions, that are mostly used in the decision-making layers, should be now provided. These functions are:

Sigmoid Activation Function:

The Sigmoid Function is, as its name announces, an "S" shaped diagram where every input value is mapped to an output between zero and one. The more positive the value in the input is, the closer it will be to the output 1, or the more negative the value, the closer it will be to 0, respectively. This is due to the mathematical model that describes the Sigmoid Function that is shown below:

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

Graphically, the Sigmoid function is an "S" shaped line that goes like this:

Figure 18 Sigmoid Activation Function

Sigmoid is commonly used for models where we have to predict the probability as an output. It is the ideal option to use when classifying between two case scenarios since the probability of any outcome exists only between the range of 0 and 1, and the Sigmoid can choose which case is more likely to be represented.

Softmax Activation Function:

Now, if you have to classify between more classes at the classifying layer, Sigmoid is not going to be useful. Softmax, on the other hand, is a combination of multiple Sigmoid functions. It calculates the relative probabilities and returns the probability of each class. So, the present function is mainly used as an activation function for the final layer of the neural network, ideally when dealing with multi-class classification.

Mathematically it is expressed as:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

Tanh Activation Function:

Tanh function is very similar to the sigmoid activation function and has a similar "S" shaped design but more vertically extended. Its range is between -1 and 1 and this makes it more symmetrical in regard to zero than the sigmoid is, making it possible to map the output values to strongly negative, neutral, and strongly positive. Its mathematical expression is:

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

And it is graphically expressed by:

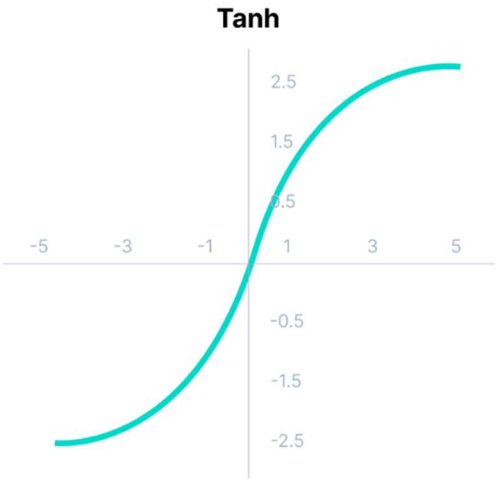


Figure 19 Tanh Activation Function

2. 4. iv *Loss Functions*

A loss function is a function that compares and quantifies the difference between the ground truth labels and predicted output values. More specifically, it measures how well a neural network captures meaningful patterns and makes accurate predictions [23]. When training a neural network, we aim to minimize this loss, so as to increase the model's accuracy and robustness.

The choice of an appropriate loss function must be carefully considered to achieve optimal training and improve the overall performance of our CNN model. So, we must choose from a range of different ones, that are to be presented below, considering the task that we are trying to manage (regression or classification). Thus, here are some loss functions that are commonly used [24]:

Mean Squared Error (MSE):

One of the most popular loss functions, MSE, calculates the average of the squared differences between the ground truth values and the predicted outputs and it is often used for regression tasks, where the goal is to predict continuous values. As mentioned before, the difference is squared, disregarding whether the predicted value is higher or lower than the target value. However, values with a large error are penalized. The Mean Squared Error loss function is described by:

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)$$

Where y_i stands for predicted output and $\hat{y}_i$ for the ground truth value.

Binary Cross-Entropy:

Binary cross-entropy is commonly used for binary classification problems, where the output is a single value to classify between two pre-set categories. It measures the dissimilarity between predicted and true labels, with the use of logarithmic terms to penalize larger errors. In binary classification, the outcome can only be between two true values of 0 or 1. Thus, to accurately determine the loss it needs to compare the actual value (0 or 1) with the probability that the input aligns with that category (p(i) = probability that the category is 1; 1 - p(i) = probability that the category is 0). This process can be represented as a single equation, which is shown below:

$$BCE = -\frac{1}{N}\sum_{i=0}^{N}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

Where N is the total number of samples in the dataset, y represents the true value and $\hat{y}$ represents the predicted value.

Categorical Cross-Entropy:

Categorical cross-entropy is used for multi-class classification tasks. It calculates the average cross-entropy loss over all classes, comparing the predicted class probabilities to the true labels. It penalizes larger differences between predicted and true class probabilities. The penalty is

logarithmic, assigning a low score for minor variances and a high score for significant differences. It is similar to Binary Cross-Entropy, but their difference is based on the number of classes that you need to classify between.

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot \log\left(\widehat{y_{ij}}\right)$$

Where N is the total number of classes in the dataset and M is the total number of samples.

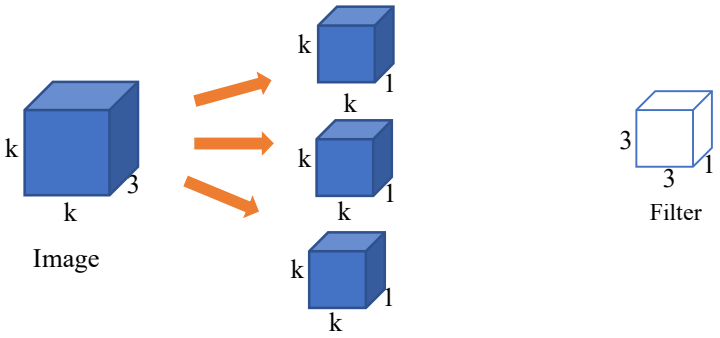
y_{ij} represents the true probability for class i and sample j.

$\widehat{y_{ij}}$ represents the predicted probability for class i and sample j.

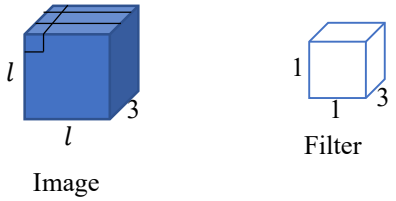
2. 5 *Xception architecture*

The Xception architecture is a version of the Inception family architecture, and its name stands for "Extreme" Inception. The idea behind this creation is the replacement of the convolution layer with a depthwise separable convolution layer, since, as proven in [6 paper], these functions can be nearly identical. The reason behind this action is efficiency and the reduction of computational cost as we will see below.

The separable convolution is an operation consisting of two individual operations: 1) a depthwise spatial convolution and 2) a pointwise convolution. So, talking about the depthwise separable convolution, imagine having a "k by k" image in the RGB color field meaning that it has a depth of 3 layers as the 3 channels of RGB. Imagine again, that we want to convolute this $k \times k \times 3$ image with a filter that is of size $3 \times 3 \times 1$. This means that we must convolute every channel of the image with this filter having a total of $3 \times 3 \times 1 \times 3$ actions.



After this convolution, we end up having an $l \times l \times 3$ image (l may be equal to k depending on padding and stride of the convolution). To perform a pointwise convolution, we have to convolute the $l \times l \times 3$ image with a $1 \times 1 \times 3$ filter.



This makes a total of $1 \times 1 \times 3$ multiplied by the number of filters that we are willing to use, let's say 16. So, this number is $1 \times 1 \times 3 \times 16$, producing an image of $l \times l \times 16$. Thus, we would have: $(3 \times 3 \times 1 \times 3) + (1 \times 1 \times 3 \times 16) = 27 + 48 = 75$ calculations.

Now, if we had a normal convolution, we should use a filter that was $(3 \times 3) \times 3 \times 16$ for kernel size, channels, and the number of filters that we use. In this case, we would have $3 \times 3 \times 3 \times 16 = 432$ calculations.

As we can see from all above, the separable convolution is much more effective than the normal convolution in terms of computational cost, and by increasing the number of filters, it becomes more and more lighter than normal a convolution.

2. 6 *small-arch Xception architecture*

Based on the Xception architecture, the authors of [1], designed a model that is smaller than the original one in terms of the number of parameters and the depth of the network. This was done in order to fit and run this network to a portable machine with limited capabilities such as the Jetson series. As we saw in Fig. 3, 20 and we can see again below, the small-arch Xception has 3 parts (input, hidden layers, and the output) as the original version, but only two flows (entry flow and exit flow). The entry flow, which includes a rescaling of the image, to make the pixel values vary between 0 and 1, a convolution using 8 filters, a batch normalization, and a ReLU activation layer. Additionally, it contains a set of ReLU, Separable Convolution, Batch Normalization, ReLU, Separable Convolution, Batch Normalization, and Max Pooling layers with an external convolution skip connection of 8 filters. For the exit flow, we have a set of Separable Convolution, Batch Normalization, ReLU, and a global Average Pooling layer followed by an FC layer. So, the points that these two differ are:

1) The number of filters used at the entry flow convolutions.
2) The times that this set of entry-level layers are iterated (1 instead of 3)
3) The lack of middle flow where a set of ReLU, Separable Convolution, and Batch Normalization is repeated many times with a large number of filters.
4) The lack of ReLU, Separable Convolution, Batch Normalization, and Max Pooling layers in the exit flow.

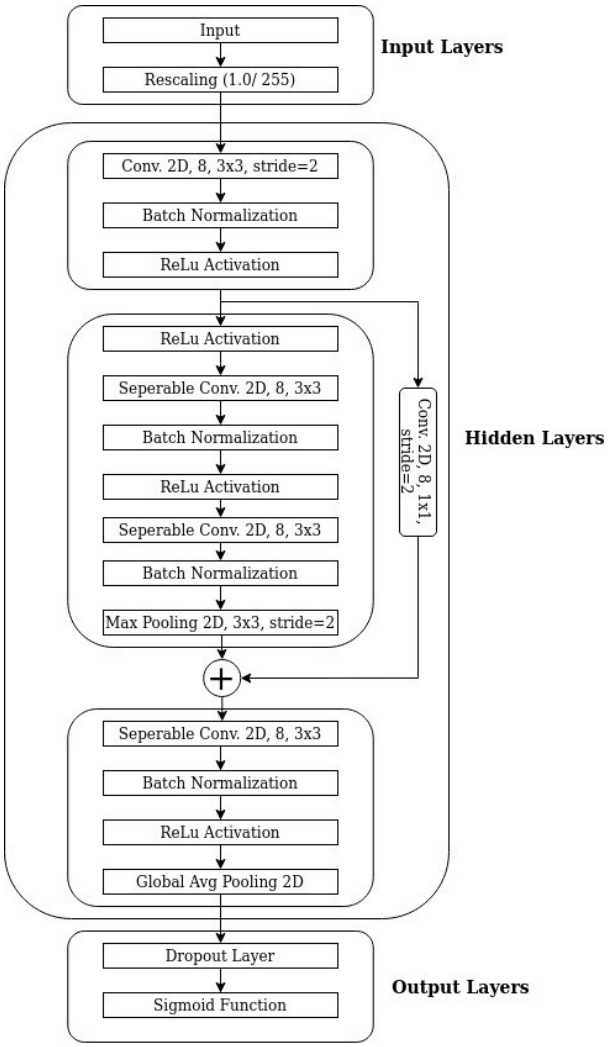


Figure 20 small-arch Xception

2. 7 *ShRe-Xception architecture*

Now, the differences between our model and the small-arch Xception model are going to be explored. In general, we aimed to take the latter model and expand it in such a way that it ends up a step closer to the original version. Hence, our model is a combination of these two previous models. Two of the flows are used like the authors of small-arch Xception did, as the middle flow with a large number of filters iterative process will explode the network's computing cost. However, there are two changes that the network has gone through, and they are:

1) The addition of an extra set of ReLU, Separable Convolution, Batch Normalization, Max Pooling, ReLU, Separable Convolution, Batch Normalization, and Max Pooling layers but with the use of 16 filters instead of 8 used in small-arch Xception.
2) The breaking of this previous set into two different ones and the addition of a skip connection containing a convolution with the same number of filters as the separable convolutions existing in the block, for every set of ReLU, Separable Convolution, Batch Normalization, Max Pooling layers.

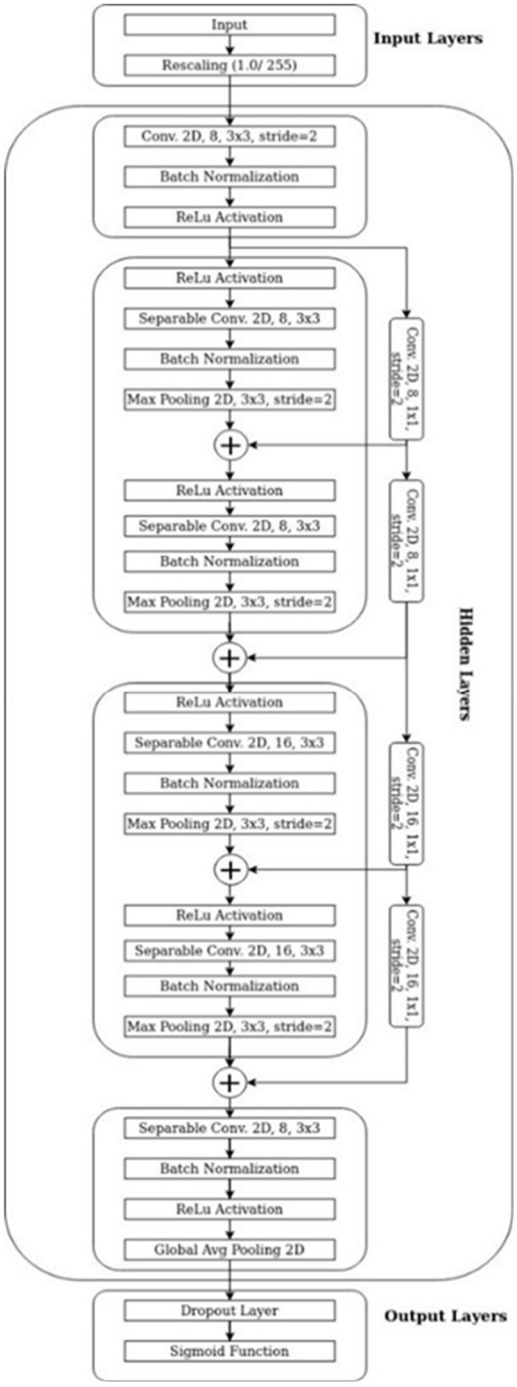


Figure 21 ShRe-Xception architecture

Still, these changes did not come randomly. To end up with this model, a lot of experiments and changes in the model's architecture were performed until one with significantly better metrics was invented.

As discussed in the Introduction section, the testing, for all the different models and all the different ways a model was trained, was performed on The FLAME and the FLAME 2 datasets.

Chapter 3 Datasets

In this work, the only datasets used are uploaded on IEEE DataPort. Everything began with finding [1] study which contained a training and a testing dataset the FLAME dataset and later, as we wanted to make the model more robust, the FLAME 2 dataset [25] was discovered, containing a summary of both RGB and IR images. The latter dataset was used to inject our already trained model with more RGB images via transfer learning and then used IR images to make the network classify images from a different spectrum too.

3. 1 *The FLAME dataset: Aerial Imagery Pile burn detection using drones (UAVs)*

This compilation of images was captured with the help of UAVs, during a planned operation by a fire department and a team of scientists. In this dataset [12], there are videos and images from whom the seventh and the eighth repository are used, which include images with 254px254p resolution, which are a little biased, as more "Fire" images are existing in it. The 7th repository consists of 39375 images for the "Fire" vs "non-Fire" classification process. The 8th repository consists of 8617 frames that are used for testing purposes in our work. This dataset has been provided "ready for use" as it is already split into training and testing in an 80/20 ratio. Later, in the training procedure, the training subset was split into training and validation in an 80/20 ratio configuring a total of 64/16/20 ratio for training, validation, and testing respectively. Below, you can see some of the images that were used:

Figure 22 Fire cases in Flame dataset

Figure 23 non-Fire cases in Flame Dataset.

3. 2 *FLAME 2: Fire detection and modeling: Aerial Multi-spectral image dataset*

The Flame 2 dataset [25], likewise the previous one, contains images of "Fire" and "non-Fire" cases that were captured on a planned operation. The difference between the 2 datasets is that the latter one contains images of both RGB and IR frames. So, by using both of them, we can have a larger number of RGB images of both "Fire" and "non-Fire" cases and simultaneously, gain the opportunity to train our network to classify images using a thermal camera. This dataset contains many repositories to choose from, but only the 9th folder which accommodates 53,541 254px254p RGB/ IR frame pairs cropped and formatted to have similar field of view (FOV) and perspective, is used. There is another folder that has similar images but with 3840px2160p resolution for RGB and 1920px1080p resolution for IR images. The first one was chosen, rather than the high-resolution one, because the first dataset that was employed had the same resolution and we did not want to affect the training procedure when a transfer learning technique would possibly be performed. Below, in Figs 22, 23 some of the used images are presented.

Here, it should be noted that this dataset was not ready-to-use, due to the lack of separation into training, validation, and testing folders. Separation was done manually and an implementation of an initial 90/10 split for training and testing datasets was done followed by a split in the training subset with an 80/20 ratio for training and validation. This provided a total of 72/18/10 ratio for training, validation, and testing respectively.

Figure 24 Fire cases using RGB/IR spectrum in Flame 2 Dataset.

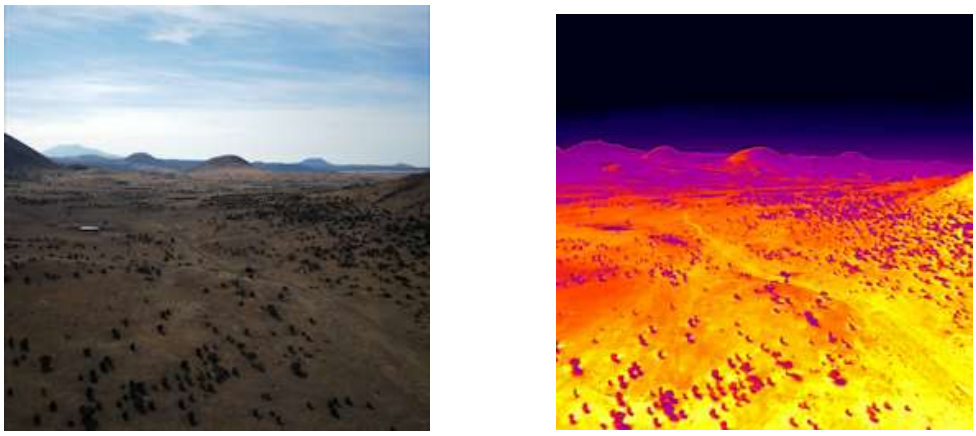


Figure 25 non-Fire cases using RGB/IR spectrum in Flame 2 Dataset.

3. 3 *Combined Dataset*

For comparison purposes, similar data for training, validation, and testing were required, but with respect to the same image spectrum. This required producing a combined dataset with all the RGB images from THE FLAME and THE FLAME 2 datasets concatenated, providing a compilation of 87,486 training and validation images and 13,965 testing images. The Infrared dataset which remains solid, will be used as a secondary base of comparison and that will be discussed even more in the next sections. Below, it is shown a summary of the containment of every dataset in this work.

Table 1 Number of images per dataset

Number of Images per Dataset	Types of Images			
	RGB Training and Validation	RGB Testing	IR Training and Validation	IR Testing
Flame	39375	8617	-	-
Flame 2	53541	5348	15637	1739
Combined	92916	13965	-	-

Chapter 4 Metrics

After training every network, the evaluation process started. This process is necessary so as to realize how accurate and robust the network that is employed is. In our work, this is done by evaluating all the models both on the combined and the IR testing sets. The testing procedure's output is a combination of the loss and the accuracy of the model during the testing phase. From these two results, the performance of the model can be assessed, but in order to test in depth the evaluation power of the produced models and compare them equally, the enhancement of some other metrics is necessary for the comparison procedure. Thus, these are some of the metrics that are used in our work:

4. 1 *Confusion Matrix*

The Confusion Matrix is, as its name announces, a matrix that shows the performance of the model in classification problems. When dealing with a classification problem the resulting decision is usually a binary one and there are 4 possible results. True Positive (TP) indicates the model predicted the input as true when the actual observation was true. False Positive (FP) indicates the model predicted true, but the actual observation was false. False Negative (FN) indicates the model predicted a false, while the actual observation was true. True Negative (TN), which indicates the model predicted a false, while the actual outcome was also false, as can also be seen in Fig. 24 below.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 26 Confusion matrix [26].

To obtain the values for the confusion matrices four counters were set during classification testing which were changing one of the four values of the above confusion matrix after the network had decided about the content of the image that passed through.

After calculating the confusion matrix values, we can calculate some other metrics that are directly dependent on those values [27].

4. 2 Precision

When working on a dataset that has an imbalanced class distribution, the accuracy is not enough to determine if your model is robust enough. Precision is a metric that calculates the ratio of how many positive outcomes the network recognized correctly in relation to how many positive outcomes did the network recognized in total. In more detail, the equation that calculates the Precision of the network is:

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

4. 3 Recall

Recall or Sensitivity is the metric responsible to calculate the number of correctly positive predicted outcomes in relation to the total of actual positive outcomes. The more this value tends to 1, the more possible is for the network to correctly detect the majority of the real positive frames. More formally:

$$Recall = Sensitivity = \frac{TP}{TP + FN}$$

4. 4 F1 score

Depending on the experiment we go through, sometimes higher priority is given to one of the two previous values. Still, there are many more applications that both of them are essential. F1 Score is the metric that combines Precision and Recall in a mathematical equation which is:

$$F1\ Score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$$

A high F1 Score means a noticeably great value to both Precision and Recall and a good balance between them. However, when it is not high enough, it is a little bit complex to understand the reasons behind it, but you can certainly look for the two metrics on which it depends.

Chapter 5 Experimenting Process

Before getting deeper into the experimenting process, a report should be made containing information on what kind of hardware is used for these tryouts to be performed. More specifically, this is a brief description of the computing power that is used as well as the libraries that are included in the working environment.

Epigrammatically, for this work, a laptop computer with an i7 intel core was put into service and performed all the training and testing procedures. The laptop has 16 GB of RAM space and no other external resources like GPU or additional RAM were used in order to perform the procedures that are to be mentioned later in this section. Thus, every machine with those capabilities can repeat the steps and confirm the results of the experiments.

As for the software that is used, all the code is written in Python 3.6.9 and compiled into a virtual environment so as to isolate only the necessary libraries to perform these tasks. In the virtual environment it is installed:

1. Tensorflow 2.3.0 and Keras 2.4.0
2. Cv2 (known as opencv)
3. tqdm
4. scipy
5. pickle5
6. numpy
7. random
8. itertools
9. scikit-image
10. matplotlib.pyplot

5. 1 *small-arch Xception.*

As seen before small-arch Xception is a smaller version of the original Xception network, used in [1]. To practice on neural networks, the small-arch Xception architecture was recreated in our system. The FLAME dataset was downloaded and as it was already split into training, validation, and testing sub-folders, the training procedure started. The training lasted for 40 epochs and the one giving the best results was the 25^{th} model, identical to this that the author announces.

After training small-arch Xception, the testing folder contained in THE FLAME was employed to test our model. Following the authors' results, when tested, it accomplished an accuracy of 68.72% (see Table 3). This metric is slightly different from the one that the authors published (76.23%), but, to our knowledge, this is something that occurs when using a stochastic model and the training procedure is performed on machines with different hardware (CPU was used for training instead of GPU). For testing the model with the best results mentioned before was evaluated.

Below, the training procedure of the small-arch Xception on the FLAME dataset is presented.

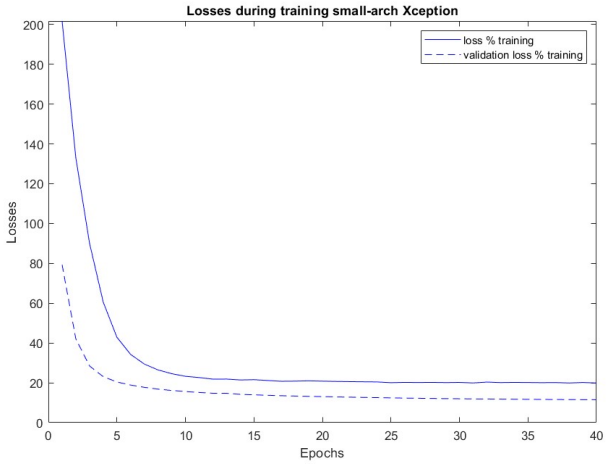


Figure 28 Losses during training small-arch Xception

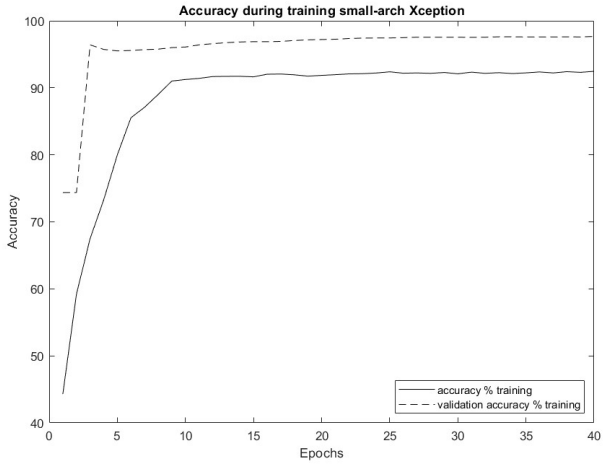


Figure 27 Accuracy during training small-arch Xception

For comparison purposes, the combined dataset was produced. To compare the models that we are going to experiment with equally, the accuracy presented above will not be used. The model will be tested on the combined dataset, producing the base model from where the experimenting journey will start. Further than the discussed accuracy, more metrics were employed to test the models' robustness, as we will discuss further in the results section.

5. 2 *Transfer Learning*

When testing the small-arch Xception, a new dataset was discovered containing aerial images of wildforests containing "Fire" and "non-Fire" frames. In order to enhance these images to our training procedure the transfer learning technique was employed. As discussed before, THE FLAME 2 dataset contains both RGB and IR images of this theme.

Firstly, the network was injected with RGB-only images via transfer learning. To apply this technique, all input and hidden layers were frozen and only the classifier's layers were trained. After that, infrared-type images were inserted in the training procedure, again with the use of transfer learning, in order for the designed model to classify both RGB and IR images of forest fires, expanding our system's vision. This retraining procedure with infrared images was applied to both the base model (small-arch Xception trained with THE FLAME) and the latest produced model (small-arch Xception retrained with RGB images of FLAME 2), producing two comparable models. In the Result section, there will be a measurement of how transfer learning affects the ability of the model to learn more information. For now, the evolution of the training process is presented below.

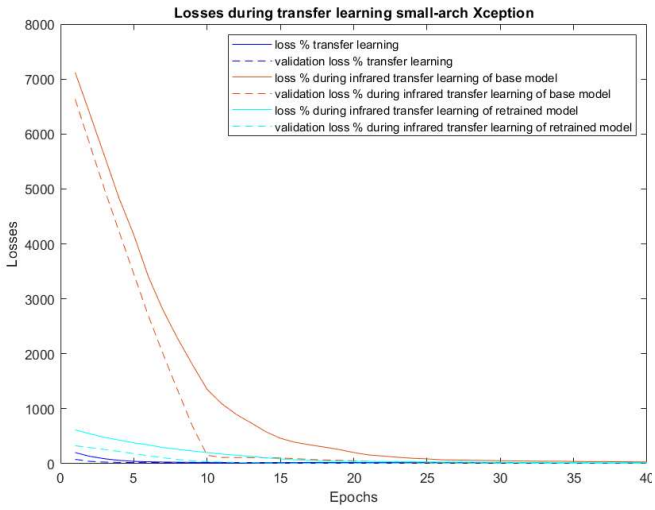


Figure 29 Losses while transfer learning training.

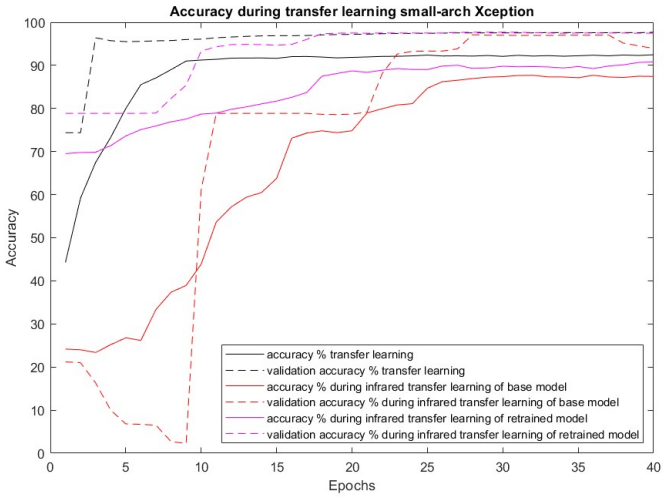


Figure 30 Accuracy while transfer learning training.

5. 3 *Training from scratch*

After that, a network that can classify images of both datasets with a specific accuracy value is produced. But in fact, that happened by using transfer learning. This technique is very helpful in cases where an empowerment of a model is needed but this can maybe cause a little drawback. In your try to teach your neural network to classify images of a new additional dataset, there is a danger for your network to forget or to decrease its capability to recognize images that had been learned at the beginning of the process.

Thus, an experimenting stage where an effort to train our network with all the images from the very beginning, started. So, the combined dataset that contains all the RGB frames was employed to train the small-arch Xception. The training procedure lasted for 40 epochs again and the model with the best results was the 33rd. The produced model was evaluated on the testing set of the combined dataset. Moreover, an additional test for this model on the infrared testing set was performed to assess its capabilities on an infrared set that it is not trained on.

After these experiments, there was a need to train this new model on IR images too. So, as did before, we took the model that has been initially trained with all the RGB images and retrained it via transfer learning with thermal images of THE FLAME 2 dataset. Lastly, this recently produced model was tested on both the combined RGB and the IR testing sets, so as to have a fair comparison between these two and the other two models that were produced in the previous subsection.

Below, training evaluations for small-arch Xception trained with all RGB images from the beginning and its infrared transfer learning reinforcement are presented.

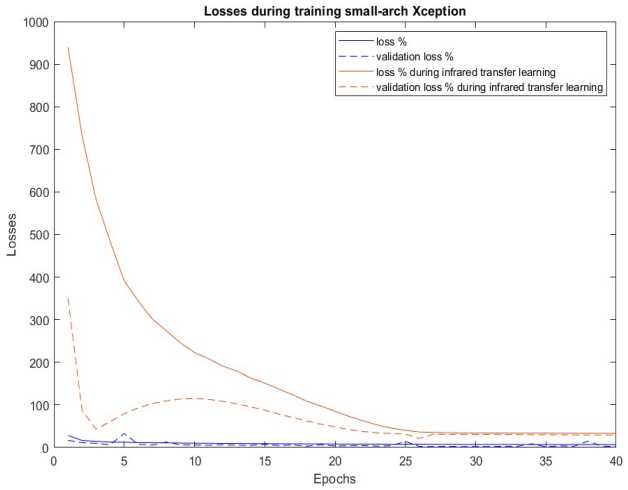


Figure 32 Losses during training small-arch Xception

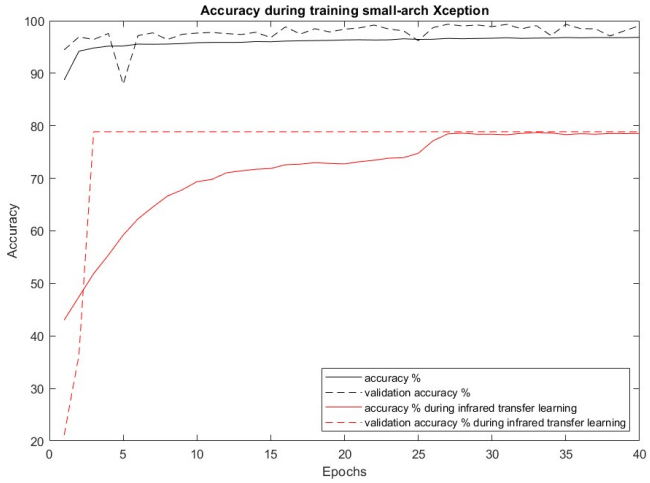


Figure 31 Accuracy during training small-arch Xception

5. 4 *The idea of ShRe-Xception*

It is a fact that, as we can see in Figs 2, 3, and 4, that our ShRe-Xception is a hybrid model produced to combine the best characteristics of the two previously created models. It remains small, in order to can be installed in machines with limited computing power, but extended as well, in both depth and complexity to be more robust in terms of classifying wildforest fire images.

To train this new network, the same procedure done before, in training the small-arch Xception with all the images from the very beginning, was followed. The network was trained with all the RGB images from the combined dataset, for 40 epochs again, and the model with the best results came to be the 24th. Except for testing on the optical spectrum, as we did before, the model's performance was tested on the infrared spectrum too. Last, transfer learning was performed for the IR spectrum to the already trained with RGB images model and the latest produced model was evaluated on both the combined and the infrared testing set.

Below the training evaluation of all experiments performed on the ShRe-Xception architecture is presented.

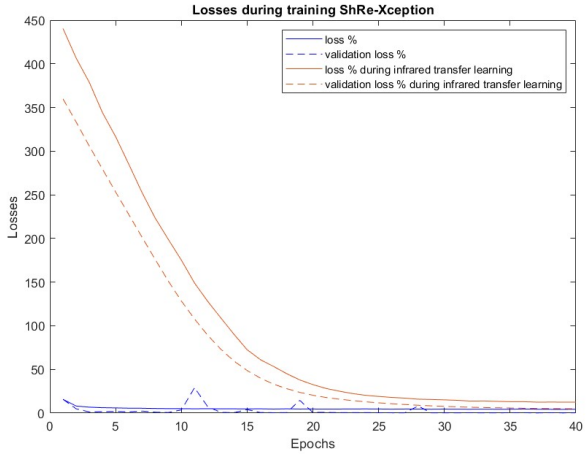


Figure 34 Losses during training ShRe-Xception

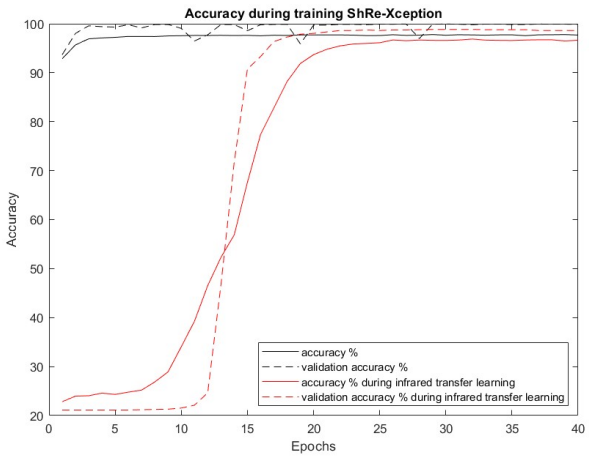


Figure 33 Accuracy during training ShRe-Xception

With this novel model architecture and the way the models were trained, on the one hand, the testing accuracy was increased on testing sets, in comparison with previous tested models, and on the other, infrared classification was enhanced. In our belief, infrared classification plays a vital role in early fire detection and can be very useful, especially in the night hours when every prediction is even more difficult to be achieved.

To conclude, we have two small-arch Xception models trained on FLAME and FLAME & FLAME 2 RGB datasets, respectively. Additionally, we have two others that were produced when injecting IR images into the previous two. A small-arch Xception that was trained with all the RGB images from the beginning with its complementary one that was retrained with IR images and the last two ShRe-Xception architecture models that were trained with RGB-only images and retrained with IR, respectively.

Chapter 6 Results

In this section, there will be a full report of all the results that came out when performing the experiments that we talked about above. All models were tested both on RGB and IR datasets and confusion matrices and values of the metrics that we used to verify every network's robustness are provided below.

6. 1 *Small-arch Xception trained on THE FLAME dataset.*

Starting from the simple small-arch Xception model the results of the testing procedure will be presented when inserted into the network all the images from the combined dataset. First, the confusion matrix is presented:

Table 2 Confusion matrix of the small-arch Xception trained on FLAME and tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	5110 (tp)	4003 (fn)
Actual negative	364 (fp)	4486 (tn)

Then, all the metrics collected from the testing process:

Table 3 Testing metrics of the small-arch Xception trained on FLAME and tested on the combined dataset.

Metrics	Model
	small-arch Xception
Accuracy	68.72%
Precision	93.35%
Recall	56.07%
F1 Score	70.06%

Additionally, the same model that has only been trained on THE FLAME dataset was tested on the IR testing set of THE FLAME 2 dataset and the results were:

Table 4 Confusion matrix of the small-arch Xception trained on FLAME and tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	291 (tp)	76 (fn)
Actual negative	46 (fp)	1324 (tn)

Table 5 Testing metrics of the small-arch Xception trained on FLAME and tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	92.97%
Precision	86.35%
Recall	79.29%
F1 Score	82.67%

The reason behind testing a model that is trained on a particular dataset, on another dataset is on the one hand to assess the effect that transfer learning has and on the other hand to check model's robustness on images that it has never dealt with previously, as this will be the task for the model to solve.

6. 2 *Small-arch Xception retrained via transfer learning.*

As the results section will follow the chronological order of the experiments, now the results of testing are presented, after performing RGB transfer learning to our previous model:

Confusion Matrix:

Table 6 Confusion matrix of the small-arch Xception trained on FLAME and retrained with RGB images of THE FLAME 2, tested on the combined dataset.

Ground truth	Predicted	
	Positive	negative
Actual positive	7580 (tp)	1533 (fn)
Actual negative	1939 (fp)	2911 (tn)

And the metrics:

Table 7 Testing metrics of the small-arch Xception trained on FLAME and retrained with RGB images of THE FLAME 2, tested on the combined dataset.

Metrics	Model
	small-arch Xception
Accuracy	75.13%
Precision	79.63%
Recall	83.17%
F1 Score	81.36%

Taking a further step, this model was tested on the infrared set, too, to have a base of comparison with the model that will be produced after the infrared transfer learning.

Confusion Matrix:

Table 8 Confusion matrix of the small-arch Xception trained on FLAME and retrained with RGB images of THE FLAME 2, tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	294 (tp)	73 (fn)
Actual negative	23 (fp)	1347 (tn)

And the metrics:

Table 9 Testing metrics of the small-arch Xception trained on FLAME and retrained with RGB images of THE FLAME 2, tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	94.47%
Precision	92.74%
Recall	80.10%
F1 Score	85.96%

As we can see from the results, the transfer learning procedure helped the evolution of both RGB and IR classification. Additionally, not only the accuracy of the testing process did increase but all the metrics seemed to increase as well, making the model appear more robust during evaluation.

The next step, as discussed before, was to train our two previous models with the infrared training set of the FLAME 2 dataset. Below, two confusion matrices will be presented and then, the metrics that came up when tested those models will be handed over, in order to have a clearer understanding of what happened after performing transfer learning with infrared images.

First, when tested on the combined RGB dataset the confusion matrices that resulted were:

Table 10 Confusion matrix of the small-arch Xception trained on FLAME, retrained with infrared images of THE FLAME 2 and tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	1407 (tp)	7706 (fn)
Actual negative	0 (fp)	4850 (tn)

Table 11 Confusion matrix of the small-arch Xception trained on FLAME, retrained with RGB images, retrained again with infrared images of THE FLAME 2 and tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	7001 (tp)	2112 (fn)
Actual negative	998 (fp)	3852 (tn)

Testing metrics:

Table 13 Testing metrics of the small-arch Xception trained on FLAME, retrained with infrared images of THE FLAME 2 and tested on the combined dataset.

Metrics	Model
	small-arch Xception
Accuracy	44.81%
Precision	100.00%
Recall	15.43%
F1 Score	26.75%

Table 12 Testing metrics of the small-arch Xception trained on FLAME, retrained with RGB images, retrained again with infrared images of THE FLAME 2 and tested on the combined dataset.

Metrics	Model
	small-arch Xception
Accuracy	77.72%
Precision	87.52%
Recall	76.82%
F1 Score	81.82%

Then these models were tested on the infrared dataset to assess what impact transfer learning has had on them:

Confusion Matrices:

Table 14 Confusion matrix of the small-arch Xception trained on FLAME, retrained with infrared images of THE FLAME 2 and tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	295 (tp)	72 (fn)
Actual negative	55 (fp)	1315 (tn)

Table 15 Confusion matrix of the small-arch Xception trained on FLAME, retrained with RGB images, retrained again with infrared images of THE FLAME 2 and tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	358 (tp)	9 (fn)
Actual negative	46 (fp)	1324 (tn)

Testing metrics:

Table 16 Testing metrics of the small-arch Xception trained on FLAME, retrained with infrared images of THE FLAME 2 and tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	92.68%
Precision	84.28%
Recall	80.38%
F1 Score	82.29%

Table 17 Testing metrics of the small-arch Xception trained on FLAME, retrained with RGB images, retrained again with infrared images of THE FLAME 2 and tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	96.83%
Precision	88.61%
Recall	97.54%
F1 Score	92.86%

After the infrared reinforcement was performed, two things were observed:

1. The first model dropped its accuracy both on RGB and IR testing sets proving that it was not ready to deal with infrared images as it was not powerful enough in the optical spectrum.
2. The already retrained model showed a greater ability to adapt to other datasets when it needs to be retrained to improve its classification performance.

6. 3 *Small-arch Xception trained initially with all the RGB images.*

As mentioned, many times, the same architecture was produced and trained, by setting as an input all the RGB images from concatenating FLAME and FLAME 2. So first, the latest produced model was tested on the combined dataset:

Confusion Matrix:

Table 18 Confusion matrix of the small-arch Xception trained on the combined dataset (training set) with all the RGB images from the beginning, tested on the combined dataset (testing set).

Ground truth	Predicted	
	Positive	Negative
Actual positive	8485 (tp)	628 (fn)
Actual negative	1486 (fp)	3364 (tn)

And the metrics:

Table 19 Testing metrics of the small-arch Xception trained on the combined dataset (training set) with all the RGB images from the beginning, tested on the combined dataset (testing set).

Metrics	Model
	small-arch Xception
Accuracy	84.86%
Precision	85.09%
Recall	93.10%
F1 Score	88.92%

And then, testing on the infrared dataset will give us the base of comparison between before and after the transfer learning procedure:

Confusion Matrix:

Table 20 Confusion matrix of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning, tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	89 (tp)	278 (fn)
Actual negative	952 (fp)	418 (tn)

And the metrics:

Table 21 Testing metrics of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning, tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	29.18%
Precision	8.54%
Recall	24.25%
F1 Score	12.64%

6. 4 *Small-arch Xception trained initially with all the RGB images and retrained with infrared images.*

In this step, transfer learning with infrared images was performed. The testing results after this process when evaluating first on the combined dataset are:

Confusion Matrix:

Table 22 Confusion matrix of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning and retrained with IR images from FLAME 2, tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	7683 (tp)	1430 (fn)
Actual negative	2815 (fp)	2035 (tn)

And the metrics:

Table 23 Testing metrics of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning and retrained with IR images from FLAME 2, tested on the combined dataset.

Metrics	Model
	small-arch Xception
Accuracy	69.60%
Precision	73.18%
Recall	84.30%
F1 Score	78.35%

Since tested the previously trained model on the combined testing set, now, it will be tested on the infrared testing set, too:

Confusion Matrix:

Table 24 Confusion matrix of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning and retrained with IR images from FLAME 2, tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	207 (tp)	160 (fn)
Actual negative	290 (fp)	1080 (tn)

And the metrics:

Table 25 Testing metrics of the small-arch Xception trained on the combined dataset with all the RGB images from the beginning and retrained with IR images from FLAME 2, tested on the infrared testing set.

Metrics	Model
	small-arch Xception
Accuracy	74.09%
Precision	41.64%
Recall	56.40%
F1 Score	47.91%

From this experiment, it becomes understood that a model that is strongly trained to focus on classifying RGB images shows higher performance on the RGB testing set. Unfortunately, because this model is trained unilaterally, it cannot classify images from the infrared dataset with a decent performance.

When retrained with infrared images, as was expected, the RGB accuracy dropped but the accuracy for infrared testing appeared to rise. This means that if there is a need for the model to do both of these two classifications, the right point of training where the best results of every spectrum meet must be established, or else, maybe the best model for every spectrum should be employed to obtain the best possible outcome.

6. 5 *ShRe-Xception trained with all the RGB images.*

ShRe-Xception is our currently produced architecture. As done before, taking all the RGB images from the combined dataset and inserting them into ShRe-Xception's training procedure, created the first model. Similar to all previous experiments, this model was tested first on the total RGB testing set and then on the infrared one.

Confusion Matrix:

Table 26 Confusion matrix of the ShRe-Xception trained on the combined dataset with all the RGB images, tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	8239 (tp)	874 (fn)
Actual negative	507 (fp)	4343 (tn)

And the metrics:

Table 27 Testing metrics of the ShRe-Xception trained on the combined dataset with all the RGB images, tested on the combined dataset.

Metrics	Model
	ShRe-Xception
Accuracy	90.10%
Precision	94.20%
Recall	90.40%
F1 Score	92.26%

While getting those encouraging results from the previous experiment, an experimenting process for infrared testing began, to assess network's power on the thermal spectrum.

Confusion Matrix:

Table 28 Confusion matrix of the ShRe-Xception trained on the combined dataset with all the RGB images, tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	357 (tp)	10 (fn)
Actual negative	2 (fp)	1368 (tn)

And the metrics:

Table 29 Testing metrics of the ShRe-Xception trained on the combined dataset with all the RGB images, tested on the infrared testing set.

Metrics	Model
	ShRe-Xception
Accuracy	99.31%
Precision	99.44%
Recall	97.27%
F1 Score	98.34%

As we can see from the obtained results, this model can be used for classification in both spectrums. It is highly accurate and without being trained in IR images, it has very strong classification results and makes a great impression.

6. 6 *Already trained ShRe-Xception, retrained with infrared images.*

As this study is coming to an end, the only thing missing is a model trained to do the infrared classification better. Thus, the already efficient ShRe-Xception model was retrained with infrared images of the FLAME 2 dataset. The model that has been produced was tested in both the combined and the infrared testing sets and the results can be found below, respectively.

Confusion Matrix:

Table 30 Confusion matrix of the ShRe-Xception trained on the combined dataset with all the RGB images and retrained with IR images from FLAME 2, tested on the combined dataset.

Ground truth	Predicted	
	Positive	Negative
Actual positive	9016 (tp)	97 (fn)
Actual negative	4589 (fp)	261 (tn)

And the metrics:

Table 31 Testing metrics of the ShRe-Xception trained on the combined dataset with all the RGB images and retrained with IR images from FLAME 2, tested on the combined dataset.

Metrics	Model
	ShRe-Xception
Accuracy	66.44%
Precision	66.26%
Recall	98.93%
F1 Score	79.37%

For testing in the thermal spectrum, we have:

Confusion Matrix:

Table 32 Confusion matrix of the ShRe-Xception trained on the combined dataset with all the RGB images and retrained with IR images from FLAME 2, tested on the infrared testing set.

Ground truth	Predicted	
	Positive	Negative
Actual positive	356 (tp)	11 (fn)
Actual negative	3 (fp)	1367 (tn)

And the metrics:

Table 33 Testing metrics of the ShRe-Xception trained on the combined dataset with all the RGB images and retrained with IR images from FLAME 2, tested on the infrared testing set.

Metrics	Model
	ShRe-Xception
Accuracy	99.19%
Precision	99.16%
Recall	97.00%
F1 Score	98.07%

Unfortunately, this is our latest produced experiment and even though the model was retrained with IR images, the accuracy of the IR testing is staying still if not decreasing a little bit. As expected, on the other hand, the accuracy of the RGB testing decreased a lot and this is not a model that can be used for this kind of image spectrum.

Chapter 7 Comparison

In this section, all the results are going to be put down to comparison to assess if our experiments did or did not help the production of a real robust network. More specifically, a summarized table will be provided where all the models will be presented side by side, along with their accuracy scores on both the combined and the infrared testing set. Later, there will be a discussion about how each change affected the evolution of our network.

Table 34 Accuracy metrics for all the models tested.

Accuracy Testing	Models							
	Small-arch Xception trained on the RGB set of FLAME	Transfer learning on Small-arch Xception, retrained with RGB images	Small-arch Xception retrained with Infrared images	Transfer learning on Small-arch Xception, retrained with Infrared images	Small-arch Xception initially trained	Small-arch Xception initially trained and retrained with Infrared images	ShRe-Xception initially trained	ShRe-Xception, retrained with Infrared images
Combined Dataset	68.72%	75.13%	44.81%	77.72%	84.86%	69.60%	**90.10%**	66.44%
Infrared testing set	92.97%	94.47%	92.68%	96.83%	29.19%	74.09%	**99.31%**	99.19%

7. 1 *The transfer learning effect*

The discussion will follow a chronological order, and at first, the effect of performing transfer learning with RGB images from THE FLAME 2 dataset will be analysed. So, as can be seen, there is an increase of 6.41% in the accuracy of the RGB testing set and a 1.5% increase in the accuracy of the infrared testing set. Eventually, the retraining procedure had a positive outcome for both RGB and the IR spectrum and that's something optimistic.

The next step was to retrain both the base model and the RGB retrained one with the IR images. Thus, starting from the base model and retraining it only with IR images, a decrease of 23.91% in the RGB field was perceived. A significant drop but somehow logical, because retraining with IR images, decreases the ability of the network to classify in RGB field, as it changes the weights of the classifier layers. In contrast to this, the fourth model that had been retrained with extra RGB images and is now retrained with IR images shows greater adaptability. Talking with numbers, in the color field, the current model, not only did not drop its accuracy but increased it with an additional factor of 2.59%. In the thermal spectrum, a rise of 2.36% was achieved in the accuracy metric, as expected when injecting thermal images in the training procedure.

7. 2 *Starting from the base.*

Starting again from the beginning, the same architecture was built and trained with all the RGB images. A comparison should be conducted between this model and the second model that has been trained with all the RGB images in two stages, which is the closest version to the one we

discuss about. So, the testing process gave us an 84.86% accuracy in the color field. In comparison with the 75.13% that the second model had, it notes a 9.73% better accuracy percentage for classifying optical images of forest fires. Unfortunately, in the thermal field, the testing accuracy seems to be a lot lower than every other accuracy in the testing process. More specifically, a percentage of 29.19% was noted, making the comparison with the 94.47% of the second model meaningless. Just for mathematics, a drop of 65.28% was detected.

Now, transfer learning is performed using infrared images. The new model was down to comparison to the exact previous model that was discussed before, and this is what occurred. From 84.86% in the RGB field, the accuracy dropped to 69.60% noting a decrease of 15.26%. In the IR field, the accuracy went from 29.19% to 74.09% noting an outstanding increase of 44.9%. Looking at it only as an accuracy percentage the 74.09% is not a very high value, but the fact that these images contributed this much to the result makes the experiment successful.

7. 3 *Changing the architecture.*

Last but not least, the attention is focused on our newly produced model, the ShRe-Xception model. After training it on RGB images only and testing it on both the combined and the infrared testing set we can compare it with the second and the fifth models to assess its dynamic as an architecture. The ShRe-Xception noted an accuracy of 90.10% for the combined dataset, leaving behind the other two models with 75.13% and 84.86% accuracy, respectively. Mainly focusing on the fifth model, which has the same training procedure and only differs in the architecture, the ShRe-Xception has an increase of 5.24% just by changing the structure of the network a little bit. However, the most fascinating fact is that while it hits this accuracy in the color field, it doesn't fail in the IR field. In fact, not only it does not fail but hits an accuracy score of 99.31% which is higher than ever before. In comparison with the previous two, it has a 4.84% increment in relation to the second one and a 70.12% increment in relation to the fifth one.

Additionally, the ShRe-Xception model was retrained with infrared images from THE FLAME 2 dataset. The latter model will be compared to the previous one that shares the same architecture. In the optical spectrum, after the infrared transfer learning, the model hit an accuracy of 66.44% noting a 23.66% decrease of power in the RGB classification. Unexpectedly, even in the IR spectrum, the model lost some of its high accuracy rate. More specifically, from 99.31% it decreased to 99.19%, which is not a great loss, but we focus on the fact that a thermal training set did not help the model be more robust when tested on infrared images.

Chapter 8 Conclusion

Looking at the previous sections, the results confirmed that the ShRe-Xception architecture outperformed the small-arch Xception architecture and that the RGB-trained model scored better classification results in both the combined and the infrared dataset. On the other hand, it is a little larger as an architecture, meaning that it needs more time to be trained than the other one, but this is a process that happens only one time and it is not such a drawback.

8. 1 *Other Works*

There is a lot of research that has been published around the same task that this work is going through and has achieved encouraging results regarding successful early fire detection. In the previous study [28], a whole system of detecting home fires and contacting the fire brigade has been designed, similar to the proposed architecture. The authors use CNNs and in particular, GoogleNet to classify images in "Fire" and "non-Fire" cases. Additionally, they do not seem to do any segmentation or localization in the image to figure out the fire's position. This is not something questionable in home-fire situations, since if a fire breaks out in a house the fire department should be called anyway. Still, it could be useful information to support more firefighters' duties. In contrast to our work, the two datasets that the authors use, do not contain much of forest fire cases. Datasets mainly contain fires in the streets and buildings rather than forestry areas that our work is focusing on, but we must admit that their "non-Fire" images are tough to be classified as they contain very vivid sunsets and lighting that may confuse a lot of AI mechanisms. As last, it should be noted that this work did not include any images or detection framework for the infrared spectrum, something that can play a vital role in fire detection at early stages, from our perspective.

In another study [29], many novelties were proposed regarding the preprocessing of the input images and the utilization of the hyperparameters. First of all, a great step was made in the way that the image reaches the input layer of the network. The authors are running a segmentation procedure first, isolating a part of the image where they believe the fire is most likely to be. Then, they take this part and train/test the network, whatever the case, with this only part. This technique helps remove the background noise and all the busy environment an image can have, empowering the network's ability to recognize "Fire" and "non-Fire" cases regardless of the background environment. Then, authors follow the same path and pass the image to a CNN to do the classification task, by using an AlexNet model. In the training procedure of this particular model, a process of optimizing the model's hyperparameters has been undertaken. These parameters include the learning rate, the batch size, and the pooling. As for the learning rate, authors are experimenting with different values through training the model and end up with the one that trained best their model. Same for the batch size, after an experimenting procedure when trying different values for this variable, they end up with the best one. As for pooling, authors not only tried different ways of pooling such as average and max but made another novelty by inventing an adaptive pooling method which is a way to combine max and mean pooling. However, in that previous study, neither infrared images were used nor aerial ones with panoramic viewing angle. Moreover, we did not manage to find the latest model that is proposed in total. There are experiments presented separately for every parameter and there is not a clear final one that is proposed with the selected characteristics. To conclude, this work

sets an inspiring base for our work to evolve. Our work was focused on training experiments and model architecture and did not include optimizing the model.

Lastly, there was another study [30] where many different architectures were down for evaluation and comparison. This study presents many good results regarding some of the models' accuracy that outperform our architecture. However, it should be noted, that no other metric was presented and the images that were used, were neither aerial nor with panoramic viewing angle, mainly including frames with different background environments such as houses, roads, and working fields rather than focusing on forestry areas. Additionally, there is not any reference to IR image classification and how this system will be portable to be able to work in real action.

In contrast to all the above, ShRe-Xception and all the general study around this proposed architecture sets a base for a huge evolution. It combines aerial imaging, forestry area surveilling, RGB & IR classification for day and night patrolling and its architecture can be installed in microcontrollers that are portable to any machine that serves the purpose of saving our environment.

8. 2 *Limitations*

Deep Learning is a science that requires strong computational power to perform experiments in real data, with models that have thousands of trainable parameters. Unfortunately, when trying to train your models to a laptop pc that has not been made to do this job, the work gets even more difficult. Furthermore, the structure of the models and the large size of the dataset is demanding significant time to complete the training process. As a result of this, the margin for error was really small, because a failed training phase could mean hours of waiting.

Additionally, it should be noted that for training and testing only THE FLAME and THE FLAME 2 datasets were used. This means that models' behaviour and robustness have not been examined in other datasets or on live feed. This is very important as to have a powerful model it must be able to run unaffected by where the input images are coming from.

Moreover, these experiments have been going through only on my computer. The models have not been loaded to another computer or more ideally, either to an embedded computer or a microcontroller. This step will be critical, as our best model has been designed to work in machines with limited capabilities to be as portable as possible. Thus, if this works as expected, it gives us a signal to optimize our model more and explore the limits of its capabilities.

8. 3 *Future Work*

To be fair, the majority of the future work is based on the limitations that have been discussed in the previous subsection. Still, some solutions to our problems that will be mentioned, have already been in progress and our thoughts will be discussed in regard to the evolution of this work.

First of all, a novel architecture was produced that seems to give better evaluation results than the small-arch Xception architecture. However, can we optimize our architecture more? This is a question that can be answered only after deep experimenting and fine-tuning the thousand

training parameters of the model along with the loss functions and the optimizer. So, one of the most significant works in this study is to maximize this architecture's evaluation metrics as much as possible. Additionally, since we assessed transfer learning techniques results, an experiment where the ShRe-Xception architecture will be trained with RGB images in two stages. This process may offer even better results regarding model's testing evaluation.

Secondly, as previously discussed, the models have been trained and tested only on FLAME and FLAME 2 datasets. What should follow is to find more datasets of the same topic to test the models on or to retrain the models via transfer learning. The best-case scenario is to connect a real camera as input to the network. This will be the real test for the network to prove its power since the frames of the camera may have different angles or the camera may be at different heights and the models might not be able to correctly classify the images. In this step, other various images can be used to train our network more or to build our own dataset where the surrounding environment will be as close as possible to the real cases that we want to classify.

The last stage, that closely linked to all the previous, is to load the model on a Jetson Series controller and take as input live feed from an external camera. The execution of this process has already started because in order to promote a network that classifies aerial images, the network must be loaded to a machine that can be carried on a UAV. In this scenario, our system's capabilities can be explored more and features such as time of flight, height of accurate classifying, travel velocity and many others can be identified.

Chapter 9 References

[1]. A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Fulé, E. Blasch, “Aerial imagery pile burn detection using deep learning: The FLAME dataset.” Computer Networks. vol. 193, pp. 1-11, 2021.

[2]. S. Malinga, “Drones to the rescue at WCape emergency services”, Itweb, 2021, accessed at 03/07/2023.

[3]. Unmanned aerial vehicle - Wikipedia

[4]. Drones in wildfire management - Wikipedia

[5]. Lyndred visible vs thermal VISIBLE vs. THERMAL DETECTION: Advantages and Disadvantages (lynred-usa.com)

[6]. S. Z. Nielsen, R. Gade, T. Moeslund, H. Skov-Petersen, “Taking the Temperature of Pedestrian Movement in Public Spaces”, Transportation Research Procedia 2, pp. 660-668, 2014.

[7]. Norden Advantages and disadvantages of Thermal Imaging Camera Advantages and disadvantages of Thermal Imaging Camera (nordencommunication.com)

[8]. F. Morgado, “Thermal Imaging in AI”, nilg.ai, 2020, accessed at 03/07/2023.

[9]. B. Mesnik, “Detection, Recognition, and Identification – Thermal vs. Optical IP Camera”, Kintronics, accessed at 03/07/2023.

[10]. C. M. Bishop, "Neural networks and their applications", Review of Scientific Instruments, vol. 65, pp. 1803-1832, 1994.

[11]. F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251-1258, 2017.

[12]. A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Fulé, E. Blasch, “THE FLAME DATASET: AERIAL IMAGERY PILE BURN DETECTION USING DRONES (UAVS)”, IEEE Dataport, 2021, accessed at 09/05/2023.

[13]. K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," in IEEE Access, vol. 6, pp. 18174-18183, 2018.

[14]. P. Li, W. Zhao, “Image fire detection algorithms based on convolutional neural networks”, Case Studies in Thermal Engineering, vol. 19, pp. 100625, 2020.

[15]. L. Wonjae, K. Seonghyun, L. Yong-Tae, L. Hyun-Woo and C. Min, "Deep neural networks for wild fire detection with unmanned aerial vehicle", IEEE International Conference, on Consumer Electronics (ICCE), pp. 252-253, 2017.

[16]. F. Chollet, “Image classification from scratch”, Keras, 2020, accessed at 09/05/2023.

[17]. A. Wolfewicz, “Deep Learning vs. Machine Learning – What’s The Difference?”, Levity, accessed at 04/07/2023.

[18]. Y. Gavrilova, “Convolutional Neural Networks for Beginners”, Serokell, accessed at 04/07/2023.

[19]. J. Browniee, “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks”, Machine Learning Mastery, accessed at 04/07/2023.

[20]. Editorial Team, “Introduction To Pooling Layers In CNN”, Towards AI, accessed at 04/07/2023.

[21]. P. Mahajan, “Fully Connected vs Convolutional Neural Networks”, Medium, accessed at 04/07/2023.

[22]. P. Baheti, “Activation Functions in Neural Networks”, V7 Labs, accessed at 04/07/2023.
[23]. A. Oppermann, “How Loss Functions Work in Neural Networks and Deep Learning”, Built In, accessed at 04/07/2023.
[24]. V. Yathish, “How Loss Functions Work in Neural Networks and Deep Learning”, Towards Data Science, accessed at 04/07/2023.
[25]. B. Hopkins, L. O'Neill, F. Afghah, A. Razi, E. Rowell, A. Watts, P. Fule, J. Coen, “FLAME 2: FIRE DETECTION AND MODELING: AERIAL MULTI-SPECTRAL IMAGE DATASET”, IEEE Dataport, 2023, accessed at 04/07/2023.
[26]. N. Beheshti, “Guide to Confusion Matrices & Classification Performance Metrics”, Towards Data Science, accessed at 04/07/2023.
[27]. S. Minaee, “20 Popular Machine Learning Metrics”, Towards Data Science, accessed at 04/07/2023.
[28]. K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos", IEEE Access, vol. 6, pp. 18174-18183, 2018.
[29]. Y. Wang, L. Dang, J. Ren, “Forest fire image recognition based on convolutional neural network”. Journal of Algorithms & Computational Technology, vol 13. 174830261988768. 10.1177/1748302619887689.
[30]. R. Joohyung, K. Yukyung, K. Minsuk. “Fire Image Classification Based on Convolutional Neural Network for Smart Fire Detection”, International Journal of Fire Science and Engineering, vol 36, pp. 51-61, 2022.