Analysis of Imagery Sensors and Methodologies Towards Early Fire Detection

Maria João Santos Lopes de Sousa

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Alexandra Bento Moutinho
Prof. Miguel Abrantes de Figueiredo Bernardo de Almeida

November 2017

Acknowledgments

First I want to thank my supervisors for the opportunity to develop my master's thesis on Project Fire-camp2. Taking on a challenging problem such as fire detection provided me with a learning experience that ended up exceeding my expectations.

I would like to thank Prof. Alexandra Moutinho for the close supervision over the course of this work, and for always believing in my ability to overcome the difficulties faced during this process.

To Prof. Miguel Almeida for his availability to clarify the doubts I had, and for the preparation of the experimental trials conducted in the Laboratory of Forestry Fire Studies in Lousã.

These tests were a team effort, and I thank Cláudia Pinto, and Abdelrahman Abouali from ADAI, who collaborated on setting up the experiments. To David Salvador for his collaboration and swift work under pressure in the preparation and development of the experimental tests. To Diogo Henriques for lending the multirotor used in these trials, but also for conducting the image acquisition tests at Boom Festival, which led to part of the dataset extensively used in this thesis.

I have to especially thank the people who first worked with me on this project in the Computer Vision course. To Eduardo Araújo for his enthusiasm for this topic, and to Ricardo Maia who drove the work forward when setbacks were encountered. Besides being great lab partners, more importantly, they are even better friends.

I wish to thank my friends André Varela, André Antunes, Daniela Melo and Tomás Hipólito for their support over the last few months, which helped to keep me motivated and the work on track.

To the friends who made this experience at IST worthwhile, Beatriz Arraiano, Teresa Soveral, Patrícia Branquinho, Diogo Ruivo, Joana Correia, Joana Neto and Mónica for their constant support.

To Rita Silvestre for her lasting friendship and understanding of my extended absences.

At last, I thank my family for their unwavering support throughout my life. To Jorginho, for never letting me win, catching up with you has been the thrill of my lifetime, and always pushed me to learn new things. To my parents for being the epitome of excellence, not only as professionals but in character. Distance for sure gives perspective, and I am incredibly fortunate to have them as my greatest influence.

Resumo

Devido às alterações climáticas, as condições de risco de incêndio são mais frequentes o que precipita a maior ocorrência destes fenómenos. Com a escalada da gravidade destes eventos, existe uma maior urgência na diminuição do tempo de resposta das equipas de emergência, pelo que se torna imperativa a rápida detecção de focos de incêndio. Este trabalho visa estudar diferentes tipos de sensores e explorar técnicas de visão computacional para a detecção destes incidentes numa fase inicial. Duas linhas de investigação são examinadas, para compreender os méritos e limitações de câmaras de gama térmica e visível. Para a primeira abordagem, uma heurística de segmentação de cores foi concebida para compreender a resposta do sensor e a codificação RGB da câmara. Em segundo lugar, recorreu-se a uma abordagem de sistemas inteligentes centrada no estudo de imagens da gama visível usando uma rede neuronal profunda. Dado que este trabalho está inserido no Project Firecamp2, para avaliar o desempenho de ambas as metodologias, os resultados de teste são comparados para um conjunto de cenários experimentais com enfoque no risco de incêndio num contexto de campismo. Enquanto que a abordagem com imagem térmica estende a capacidade sensorial permitindo uma detecção mais rápida, a abordagem de aprendizagem profunda com imagens em cores tem potencial para proporcionar algoritmos mais robustos e com maior generalização. A análise comparativa entre as duas demonstrou que, para uma aplicação de detecção de fogo, o desenvolvimento de algoritmos deve abranger tanto imagens térmicas como da gama visível.

Palavras-chave: detecção de fogo, imagem térmica, imagem RGB, aprendizagem profunda, segmentação de cor, imagens aéreas.

Abstract

Due to climate change, fire risk conditions are more frequent which precipitates the greater occurrence of these phenomena. With the escalation in the severity of these events, there is an increased urgency of diminishing the response time from emergency teams, so the rapid detection of fire outbreaks becomes imperative. This work aims to study different types of sensors and explore computer vision techniques to detect these incidents at an early stage. Two main lines of research are examined, to understand the merits and limitations of thermal and visible range cameras. For the first approach, a color segmentation heuristic was devised to understand the sensor response and the RGB encoding of the camera. Second, an intelligent systems approach centered on the study of visible range images using a deep neural network. Since this work is inserted in Project Firecamp2, to evaluate the performance of both methodologies, the test results are analyzed for a set of experimental scenarios with a focus on fire risk in a camping context. While the thermal imaging approach extends the sensing ability enabling faster detection, the deep learning approach with color images has the potential to provide more robust algorithms with greater generalization. Overall, the comparative analysis between the two approaches taken demonstrated that for a fire detection application the development of algorithms should encompass both thermal and visible range images.

Keywords: fire detection, thermal imaging, RGB imaging, deep learning, color segmentation, aerial images.

Contents

Acknowledgments . . . iii

Resumo . . . v

Abstract . . . vii

List of Tables . . . xiii

List of Figures . . . xv

Nomenclature . . . xvii

1 Introduction **1**

1.1 Fire Hazard Identification and Risk Assessment . . . 1

1.2 Project Firecamp2 . . . 3

1.3 Proposed Approach . . . 4

1.4 Objectives and Contributions . . . 5

1.5 Thesis Outline . . . 5

2 Data Acquisition System **7**

2.1 Hardware Architecture . . . 7

2.1.1 Raspberry Pi . . . 8

2.1.2 RGB Camera . . . 9

2.1.3 Thermal Camera . . . 10

2.2 Experimental Tests . . . 12

3 Thermal Imaging **15**

3.1 Thermal Imaging Data . . . 16

3.2 Sensor Measurements . . . 19

3.3 Adaptive Color Scale . . . 21

3.3.1 Controlled Fire Example . . . 22

3.3.2 Boom Festival Example . . . 23

3.4 Discussion . . . 24

4 Color Segmentation Heuristic **25**

4.1 Color Segmentation . . . 25

4.2 Validation Examples . . . 28

4.2.1 Straw Examples 28
4.2.2 Tent Example 29
4.2.3 Boom Festival Example 31
4.3 Results 31
4.4 Discussion 34

5 Deep Neural Networks **35**
5.1 Neural Networks for Visual Recognition 37
5.1.1 Deep Learning Frameworks 39
5.1.2 Transfer Learning 40
5.2 Convolution Neural Networks 42
5.2.1 Convolution 43
5.2.2 Fully-Connected Layer 46
5.2.3 Convolution Layer 47
5.2.4 Pooling Layer 48
5.2.5 Dropout Layer 49
5.3 Inception v3 Model 50
5.3.1 Network Architecture 50
5.3.2 Feature Extraction 51
5.3.3 Hidden Units Activation 52
5.3.4 Classifier 53
5.4 Optimization 54
5.4.1 Gradient Descent 54
5.4.2 Loss Function 55
5.4.3 Error Backpropagation 56

6 Implementation **59**
6.1 Firecamp2 Dataset 59
6.2 Data Preprocessing 60
6.3 Training 61
6.3.1 Model Retraining 61
6.3.2 Hyperparameter Selection 63
6.4 Testing Results 64
6.4.1 Straw Example 64
6.4.2 Tent Example 65
6.4.3 Drone Example 66
6.5 Discussion 67
6.5.1 Performance Analysis 67
6.5.2 Comparison of Results 68

7 Conclusions **71**

Bibliography **75**

List of Tables

2.1 Summary of the specifications of the visible range camera. 9
2.2 Summary of FLIR Vue Pro specifications. 11
2.3 Summary of the experimental tests. 14
2.4 Summary of the analyses applied to the experimental tests. 14

3.1 Sensor resolution. 17
3.2 Camera Temperature Limits. 20
3.3 Camera Parameters. 20

4.1 Segmentation thresholds. 26

6.1 Dataset division. 60
6.2 Time intervals for the first fire alarm for the experimental tests. 69

List of Figures

2.1 Image acquisition system mounted on a drone. 7
2.2 Implementation scheme with partial pinout and respective connections. 8
2.3 3D printed part to mount the camera module adjacent to the thermal camera. 9
2.4 Installation of Raspberry Pi Camera Module V2. 9
2.5 Electromagnetic spectrum. 10
2.6 On-board thermal cameras. 10
2.7 LEIF in Lousã, Portugal. 12
2.8 Experimental setup with a drone and static platform. 12
2.9 Experimental setup for the burning of a tent. 13
2.10 Images acquired with (a) FLIR SC660, (b) FLIR Vue Pro and (c) Pi Camera V2. 13

3.1 *GrayRed* color palette. 15
3.2 Fire ignition detection with a radiometric thermal camera. 16
3.3 Temperature profile for a fire ignition. 16
3.4 Comparison of thermal imaging data. 18
3.5 Temperature versus A/D Counts. 19
3.6 Evolution of the range and median raw values for a controlled fire. 22
3.7 Fire ignition detection with a radiometric thermal camera. 22
3.8 Raw measured with Flir Vue Pro from high altitude with a 5 second timelapse. 23
3.9 Raw measured with FLIR Vue Pro for community kitchen area. 23
3.10 Heat detection with a nonradiometric thermal camera. 24

4.1 Division in color segmentation classes. 26
4.2 Fire detection with radiometric camera for Test 1. 26
4.3 Fire detection with Flir SC660 (radiometric). 28
4.4 Fire detection with Flir Vue Pro (nonradiometric). 28
4.5 Perspectives with 3 different cameras. 29
4.6 Image data from the burning of camping tent. 29
4.7 Tent burning captured by Flir SC660 (radiometric). 30
4.8 Tent burning captured by Flir Vue Pro (nonradiometric). 30
4.9 Tent burning captured by RPi Camera Module. 30

4.10 Color segmentation results. 32
4.11 Color segmentation results. 33

5.1 Artificial Intelligence: applications, domains, methods. 36
5.2 Comparison of DNN models. 38
5.3 Deep Learning libraries. 39
5.4 Novel deep learning frameworks. 39
5.5 Transfer Learning scheme. 41
5.6 Example of a simple convolutional neural network. 42
5.7 Gaussian Function. 44
5.8 Filter Matrix. 45
5.9 General neural network model with an input layer, a single hidden-layer, and an output layer. 46
5.10 Example of a convolutional layer of depth 6. 47
5.11 Parameter sharing for the slices of neurons in the convolutional layer. 47
5.12 Example featuring a max pooling operation with a 2x2 filter and a pooling stride of 2. . . . 48
5.13 Dropout example. 49
5.14 Inception V3 network architecture. 50
5.15 Example of the first Inception module of Block B. 51
5.16 Sigmoid activation function. 52
5.17 ReLU activation function. 52
5.18 General backpropagation procedure for a single hidden-layer network. 56

6.1 Samples from Firecamp2 Dataset. 59
6.2 Tensorboard representation of the operations added to retrain Inception V3. 62
6.3 Comparison of models with different learning rate and batch size of 100. 63
6.4 Color images from the straw burning trial. 64
6.5 Straw burning (static): predicted scores and the ground truth over time. 64
6.6 Color images from the tent burning trial. 65
6.7 Tent burning: predicted scores and the ground truth over time. 65
6.8 Color images from the straw burning trial captured from a drone. 66
6.9 Straw burning (drone): predicted scores and the ground truth over time. 66
6.10 Performance evaluation of the testing set. 67
6.11 Comparison of the segmentation heuristic for thermal images (left) versus the Inception classification for color images (right). 68

Nomenclature

Greek symbols

δ^h	Local gradient at the hidden layer.
δ^o	Local gradient at the output layer.
μ_d	Mean value of a Gaussian distribution
σ	Softmax activation function.
σ_g	Standard deviation of a Gaussian distribution.

Roman symbols

η	Learning rate.
$\hat{y}$	Predicted output.
b	Bias of a neuron.
f	Activation function of a hidden-layer neuron.
g	Gaussian function.
h	Impulse response of a 1-D signal.
h_j	Output of a neuron in a hidden-layer.
I	Input image matrix.
K	Filter kernel matrix.
L	Loss function.
u	Output of a pooling layer.
W	Weight matrix between hidden-layer and output layer.
w	Weights matrix between input layer and hidden-layer.
x	Input signal.
y	Output signal.

z_j	Input of a neuron in a hidden-layer.

Subscripts

i, j, k	Computational indexes.
m, n	Filter indexes.

Superscripts

T	Transpose.

Chapter 1

Introduction

Fire is a concerning natural hazard on a global scale, that has a brutal impact on the environment by disturbing the climate and natural ecosystems. This phenomenon has drastic effects on communities on a social and economic level, since it can lead to the loss of lives and material damages.

Every year the Mediterranean basin is invariably under high fire risk, with recurrent wildfires that amount to a large number of hectares of area burned [1]. This has led to continued research focused on the fire incidence in this geographical zone, and what factors drive this region to be so fire-prone [2].

In the next section, we delve into the problem at hand, by first conducting a revision of other efforts made to address fire risk assessment and identification. Then, what follows is a description in brief detail of the reasoning behind the proposed approach, along with a listing of the objectives this work sets out to accomplish. At the end of this introductory chapter, an outline is presented to establish a road map for what is in store in subsequent chapters of this thesis.

1.1 Fire Hazard Identification and Risk Assessment

Fire hazard risk results of the interplay between climate and geology making these factors critical in environmental management and risk assessment. In fact, this problem has fallen under the scope of several scientific fields, and to start, it is important to frame the developments and contributions of each.

Although most fires are human-caused, their periodic recurrence is to a great extent due to cyclical processes that occur in nature. In the spring season, when humidity levels and precipitation are higher, there is a boost in plant growth. In the summer, there are often long periods of intense drought where vegetation and soil dry up making them more fire-prone.

Fire ignition and propagation potential is strongly tied to three determinant factors: fuel, oxygen, and heat. These elements develop in consequence of the aforementioned ecosystem cycles that involve vegetation growth and climate conditions. For that reason, forestry fire hazard has been extensively studied by fields related to ecology, biogeography, and geology [3].

Meteorological conditions have also been attentively examined by the field of geography, to study how land topography and geology interact with weather variations across seasons. The evolution over time of variables like precipitation, wind speed, relative humidity and temperature help to better characterize fire regimes [4, 5].

Over the years, these analyses were enriched by the technological advances of geographic information systems (GIS) and with the valuable contribution from remote sensing with satellite images. For instance, applications include fire risk assessment [6, 7], burnt area maps [8, 9], fuel mapping [10] and decision support systems [11].

From an environmental management point of view, these methods provide a criterion for developing preventive initiatives, and prioritizing resource allocation in both monitoring and emergency response stages. However, the studies mentioned until now either using statistical approaches or satellite imagery have a serious limitation. By relying heavily on long-term data, they are unable to provide actionable information in real-time for ground teams. Despite placing certain areas under high alert based on quantification of risk due to past events, these studies overlook the urgency of diminishing the delay in response time from when a fire starts until first responders attend to it.

By nature, fire occurrence is a process with a high degree of uncertainty, both on a spatial and temporal level, making it difficult to pinpoint the threat. Therefore, the immediate identification and prompt action from emergency response teams are key for minimizing its dire consequences.

Following this line of reasoning, there has been considerable interest in applying computer vision techniques to identify fire using color imaging as well as from the thermal infrared range.

Fire detection has been the object of a plethora of different approaches that propose color models for flame pixel classification using RGB [12], YCbCr [13], HSI [14] or the YUV [15] color spaces. Toreyin et al. propose a method for flame detection in color video using temporal and spatial wavelet analysis [16].

Regarding uses of thermal infrared images for the same purpose, there have been some research efforts [17, 18, 19], yet the acquisition cost of thermal imaging systems is a roadblock for the widespread use of this technology.

Despite the fact that classic computer vision algorithms register high success percentages in classification, by being based on fine-tuned heuristics they lack the generalization to be successfully applied to new datasets and different scenarios [20]. To mitigate this limitation, researchers increasingly employ computer vision algorithms in tandem with intelligent systems approaches. As noted by Cetin et al. in their review of video-based fire detection [21], from 2007 onwards most of the approaches integrate intelligent systems for classification.

A different approach to the fire detection problem has been through video-based smoke detection, which in the case of a true positive has the merit of providing a fire alarm by spotting smoke from far away distances. In this category, the methods used can be divided into three distinct groups (1) histogram-based smoke detection, (2) temporal-analysis-based techniques, (3) rule-based techniques.

Chen et al. propose a rule-based system for smoke detection based on pixel intensity, with a chromatic decision rule, deduced from analyzing the color of smoke, and a second decision rule based on diffusion that attends to spreading attributes of smoke [22].

The early-warning fire monitoring system proposed in [23] combines a two-class Support Vector Machine (SVM) classifier with luminescence maps to remove non-fire pixels, though according to the study the method is not suitable for real-time applications due to requiring excessive computational time. Still regarding video sequences, Tung and Kim introduce a four-stage smoke detection algorithm comprised by segmentation of moving regions, fuzzy c-means clustering to select smoke regions, followed by parameter extraction and ultimately a SVM classifier [24].

More recently, in 2015, a noteworthy contribution explores smoke detection using convolution neural networks (CNNs), having achieved promising results [25]. Since then, CNNs have risen to be considered the state-of-the-art in computer vision applications and there is an increased interest in applying this and other deep neural network architectures to video analysis [26].

1.2 Project Firecamp2

The Mediterranean-type climate, due to its characteristic traits, enables the development of conditions of high-ignition propensity. In this environment camping parks present an elevated risk because, on top of material damages, fire related incidents can have fatal consequences.

In this context, the combination of combustion inducing equipment and highly combustible paraphernalia associated with camping activity, can lead to fire incidents. These present a considerable danger and often require the evacuation of thousands of people. Moreover, the human presence acts as another source of potential fire causes either by negligence or risk behaviors. These factors increase fire risk and make camping sites a hotbed for the occurrence and rapid propagation of wild fires.

Although the spacious nature of this kind of setting is naturally appealing for recreational activities such as festivals, it also poses a threat both on an environmental and human safety level. Moreover, these events are usually promoted in the outskirts of major cities, or even in isolated places with lower accessibility, increasing the degree of difficulty in what is a time-sensitive emergency response action.

In that sense, securing these venues against fire hazard is paramount from an organizational and public safety standpoint, hence the timely identification of fire risk is a relevant line of study.

In this context, the portuguese project Firecamp was carried out by the Associated Laboratory for Energy, Transports, and Aeronautics (LAETA) from 2011 to 2015. The joint investigation developed by several research centers achieved some preliminary results concerning (1) combustibility of camping materials, (2) land characterization from aerial image and (3) modeling of fire spread [27].

The need for further research, led to the development of the Firecamp2 project, starting in 2015, and in which the current work is inserted in. The main objectives of this project are (1) the analysis and understanding of fire risk, (2) modeling the burning of tents and caravans, (3) devise good practices to reduce the risk of fire incidents in camping parks, and (4) development of a safety manual against fires related to camping activities.

Under the scope of project Firecamp2, this work centers on understanding thermal imaging sensory data and exploring computer vision techniques for detection of fire incidents in an earlier phase.

1.3 Proposed Approach

As seen so far from literature review, many of the research paths taken to address fire hazard revolved around three distinct objectives. First, there have been developments in long-term fire risk assessment strategies from fields mainly related with geography. Second, from the applications of conventional computer vision algorithms to color or thermal images, the central focus has been to identify fire. Moreover, the recent efforts enhanced by computational intelligence are based on the hypotheses of detecting smoke in color images. This, however, can lead to misclassifications where visually something can appear to be smoke, but is not necessarily caused by fire.

Notwithstanding, the examples given do not capture fully what is at stake in this endeavor, since they have the fundamental limitation of detecting fire either when it is visible to the naked-eye, or if visible by the detection of smoke. Instead, by attempting to identify fire using both color and thermal imaging, countermeasures to extinguish the fire can be undertaken earlier, thus minimizing fire spread.

However, when thermal imaging systems are not empowered with radiometry capabilities, that represents a significant limitation, because the image data is deprived of quantitative information. Since the images these systems capture are only able to display relative temperatures between objects in a scene, there is no measurement of the absolute temperature. Furthermore, classic computational vision algorithms are *ad hoc*, thus requiring lengthy development timeframes for the fine-tuning for a context specific task. Aiming for better generalization, the amounts of data that have to be covered increase, which limits the performance of this type of approach. Additionally, by having only information of relative temperature differences, data suffers from high inner-class variability, since heat signatures of specific situations are dependent of the scene context.

To address this issue, as mentioned previously, it is imperative to use computer vision techniques coupled with intelligent systems. In recent months the interest in using deep neural networks for pattern recognition and video classification has been on the uprise. Much of this renewed interest in computer vision applications, particularly with convolution neural networks (CNNs), came as a result of two determinant factors. First, storing data has become inexpensive and is no longer a 21st century problem. On the other hand, it is not humanly possible to perform data analysis on all data without using artificial intelligence. Second, a process as data-intensive as image processing using CNNs became viable by the parallel computation power provided by Graphics Processing Units (GPUs), which accelerate the learning and inference stages.

While GPUs provide a significant step forward, with increased complexity and depth, neural networks take extensive amounts of hours and computing power to train. This process can amount to weeks depending on the network and the difficulty of the classification problem.

Currently, several deep learning frameworks allow the opportunity to use a pre-trained deep neural network model leveraging its acquired feature extraction capability to learn a new classification task. This concept, called transfer learning, shortens the training phase tremendously because the part of the neural network responsible for the extraction of spatial features has already been optimized and maintains its generalization.

In this work, in the context of a fire detection problem, two main lines of research are explored to study imaging sensors. The first approach will focus on the characteristics of thermal cameras and its strengths and limitations. The second approach will use color images and retrain a CNN, to evaluate if employing this type of data-driven model, without prior feature extraction, is adequate for this type of application.

1.4 Objectives and Contributions

This work aims to study different types of image sensors and explore computer vision techniques and classification algorithms to detect fire incidents at an early stage. To accomplish this objective a set of experimental tests are conducted for image acquisition purposes. Since this work is developed under the scope of project Firecamp2, the trials have an emphasis on fire risk situations related to the camping activity.

The main contributions of the present work are:

- Development of a data acquisition system for capturing synchronized images from the thermal and visible range to be used in mobile aerial platforms.
- New dataset with color and thermal images (radiometric and nonradiometric) concerning fire situations associated with camping activities.
- Evaluation of the adequacy of thermal and color image sensory data for fire detection using two different methodologies.
- Assessment of the limitations of image sensory data for fire detection using static and mobile platforms.

1.5 Thesis Outline

So far, in this introduction the motivation was addressed, and the objectives of this work outlined. Following this first introductory chapter, this thesis is organized in 6 chapters.

Chapter 2 presents a data acquisition system developed for aerial image capture, with the purpose of equipping unmanned aerial vehicles (UAVs) with thermal and color imaging devices. The chapter describes the hardware architecture proposed and the communication scheme implemented. Subsequently, the main components of the system are covered in detail, with a special attention to the thermal camera employed. Finally, the experimental conditions in which the system was tested are presented, and the types of data acquired are introduced.

Chapter 3 delves into concepts related to thermal imaging, and explores radiometric and nonradiometric data. In that regard, two different cameras are presented and their respective technical and calibration parameters are studied. Additionally, an in-depth study of the sensory data of different imaging devices is undertaken, with a particular emphasis on sensor response under real-world conditions.

In Chapter 4, a color segmentation heuristic is presented to address the problem of fire detection. Contrary to the previous chapter which focused on sensor measurements, this chapter centers attention on understanding the sensor response after RGB encoding. This chapter analyzes the variability in thermal imaging data for a set of selected fire risk situations in camping parks. At the end, the results are compared and some critical takeaways are discussed.

Chapter 5 focuses on theoretical aspects of deep neural networks, an alternative approach to the problem. The chapter starts with a review of the recent developments in artificial neural networks in the context of computer vision, and frames the current landscape of deep learning frameworks. Then, the chapter progresses with the explanation of the inner workings of convolution neural networks, followed by a detailed description of the Inception V3 model. Afterwards, some network optimization aspects are covered, namely the basis of the error backpropagation algorithm and gradient descent, which dictate the choices in activation functions throughout the network, as well as the score function employed by the classifier.

Chapter 6 centers on the implementation of a transfer learning approach for a fire detection problem using the Inception V3 model. First, an augmented color image dataset built in this work is presented, along with the partitions used for training and testing. This is followed by the outlining of the preprocessing steps necessary to extract the video frames and feed the images to the network. Subsequently, the approach to retrain the model using TensorFlow is described, as well as the process of hyperparameter selection. At last, the model is tested on new data and the results are compared with the ones from color segmentation heuristic approach.

In closing, Chapter 7 summarizes the key findings drawn from the results of the different approaches, and offers some concluding remarks concerning the fire detection problem. Furthermore, in light of the opportunities the data acquired presents, suggestions are made for future work.

Chapter 2

Data Acquisition System

This chapter presents a system developed for image acquisition, empowered by thermal and visible range imaging devices, designed to be used on unmanned aerial vehicles. Section 2.1 starts with an overview of the system and its components, followed by a breakdown of the hardware connections and a high-level description of the communication scheme. This is completed with the presentation of the imaging devices employed and their respective technical specifications. Finally, section 2.2 presents the experimental conditions of the tests developed in this work, and provides examples of the data collected.

2.1 Hardware Architecture

With the purpose of acquiring thermal and visual range aerial images, an integrated acquisition system was developed to be taken onboard a multirotor (see fig. 2.1). Its principal components are:

- On-board computer - Raspberry Pi 3 Model B
- Color Camera - Raspberry Pi Camera Module V2
- Thermal Camera - FLIR Vue Pro 336 9mm

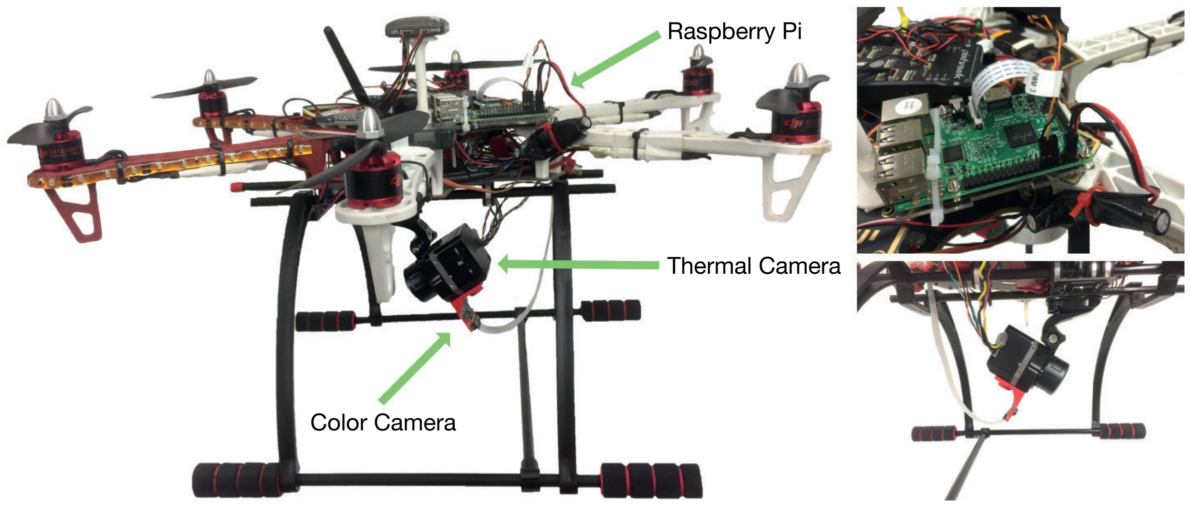


Figure 2.1: Image acquisition system mounted on a drone.

2.1.1 Raspberry Pi

The Raspberry Pi 3 Model B was selected to be the on-board computer to handle image acquisition tasks. Its 8.5 x 5.6 cm^2 size facilitates the installation on the drone frame, and with 42 g it is easy to accommodate in the drone's payload budget. This single-board computer is a widely known and extensively documented platform, used in several applications. This section focuses on aspects related to the hardware connections and describes the communication scheme implemented.

Although the thermal camera could be triggered by Pulse Width Modulation (PWM) using a flight controller like Pixhawk, for the videos from both cameras to be as synced as possible, both cameras are directly controlled by the Raspberry Pi. The thermal camera is hardware-triggered via PWM signal, and the color camera is software-triggered using the *Picamera* Python library. Figure 2.2 presents a scheme of the hardware connections for this implementation.

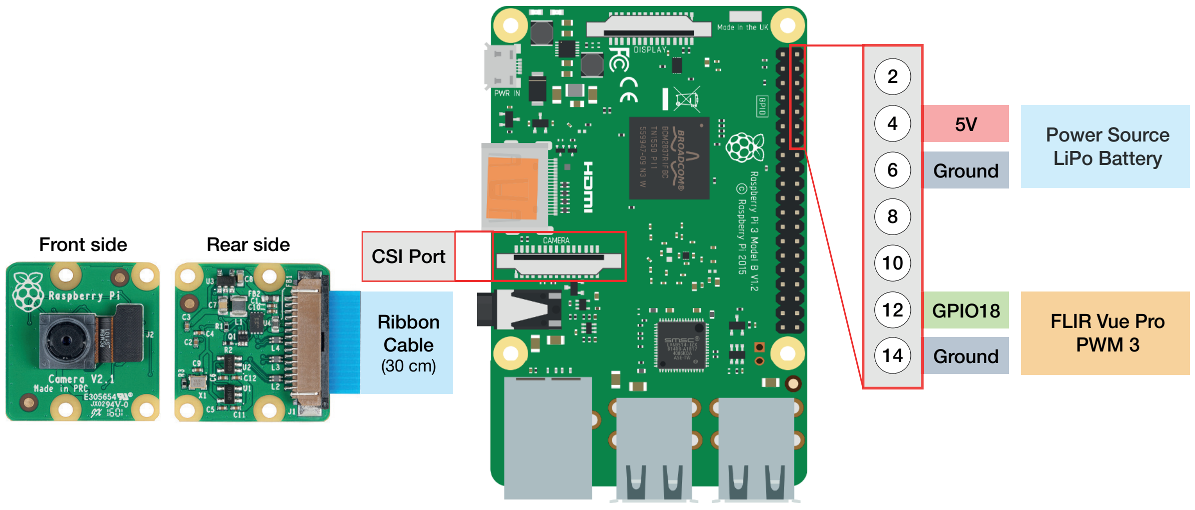


Figure 2.2: Implementation scheme with partial pinout and respective connections.

The PWM signals are sent to the thermal camera through the Raspberry Pi's General Purpose Input/Output (GPIO) Ports, namely using the GPIO 18 (pin 12), the only pin Raspberry Pi disposes capable of sending PWM via hardware. Observing the board pinout, there is a ground connection in pin 14 so the cables in the FLIR Vue Pro Accessory cable were changed in order to have the ground cable in a consecutive pin to PWM 3. The connections are visible in the upper right corner of figure 2.1.

The PWM controls have to be configured on the FLIR UAS mobile application. These were set to start recording on HIGH, and to stop recording on LOW. The PWM signal was programmed with a frequency of 50 Hz using *RPi.GPIO* Python library for a 10% and 5% duty cycle, respectively, according to the following relation:

$$\text{Duty Cycle (\%)} = \text{Pulse Width (ms)} \times \text{Frequency (Hz)} \times 100\ \%$$

To avoid interference with drone sensors, wireless connectivity is turned off and Python scripts are run through secure shell (SSH) on the ground using an ethernet cable and stopped with the same protocol upon landing.

2.1.2 RGB Camera

In order to acquire images from the visible range the Raspberry Pi Camera Module V2 was selected. This camera module allows for perfect integration with the Raspberry Pi, and due to its reduced weight it is a good choice for this application, since payload is a concern. Its main specifications are listed in table 2.1.

Table 2.1: Summary of the specifications of the visible range camera.

RPi Camera Module V2	
Size	25 x 23 x 9 mm
Weight	3.4 g
Sensor Resolution	3280 x 2464 pixels
Focal Length	3.04 mm
Horizontal FOV	62.2^{o}
Vertical FOV	48.8^{o}

Considering the difficulty that interpreting thermal imaging entails, the system will also employ the RGB camera to provide a ground truth for fire situations where it is possible to obtain visual confirmation from color images.

The camera is attached to the FLIR Vue Pro 1/4”-20 mounting hole through a 3D printed part designed for that purpose, which is illustrated in figure 2.3.

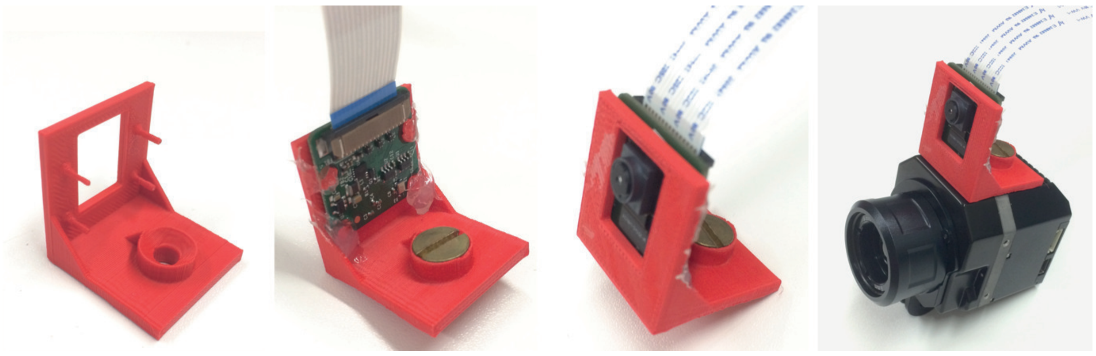


Figure 2.3: 3D printed part to mount the camera module adjacent to the thermal camera.

The camera module connects to the RPi board through the CSI camera port, (see fig. 2.4) using a 30 cm ribbon cable acquired separately.

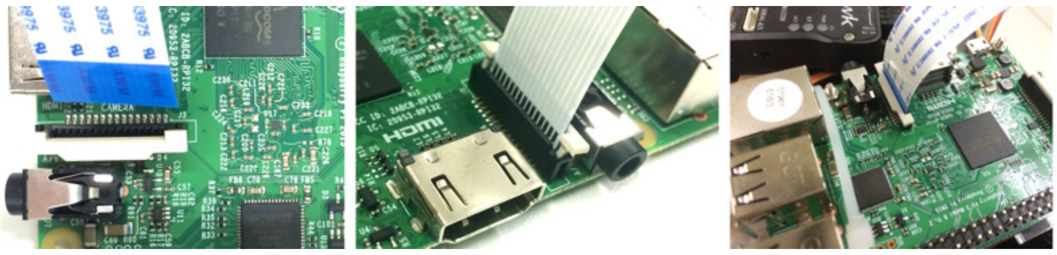


Figure 2.4: Installation of Raspberry Pi Camera Module V2.

As referenced before, on a software level the camera is triggered to record video using the *Picamera* Python library.

2.1.3 Thermal Camera

All objects above absolute zero (0º Kelvin) emit thermal infrared energy. Since fire is a source of extreme temperatures, thermal imaging sensors are a clear choice for a fire detection application.

What is referred to as thermal radiation is comprised by radiation with wavelengths in the 10^{-7} to 10^{-4} interval of the electromagnetic spectrum. Therefore, this gamma includes spectral bands in the ultraviolet, visible, and infrared region of the spectrum, as can be observed in figure 2.5.

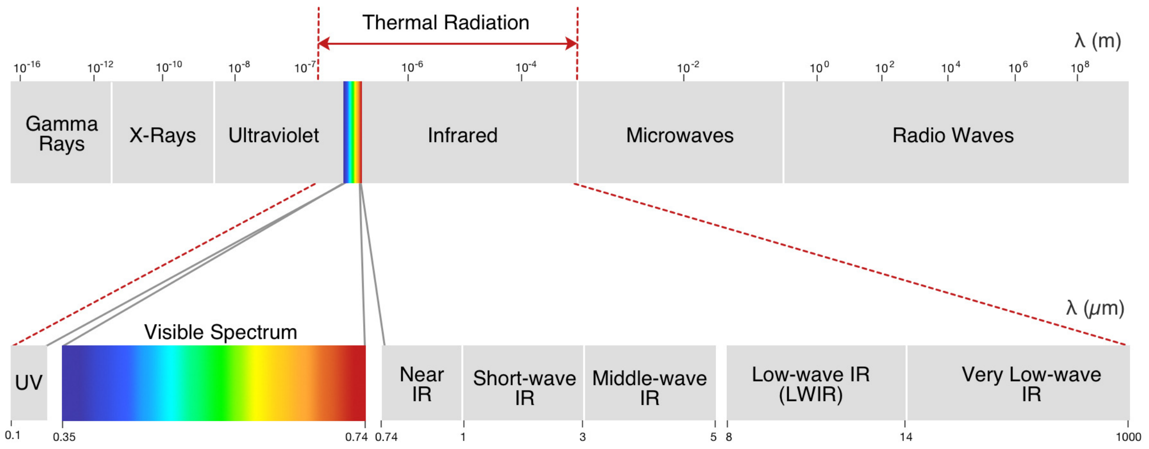


Figure 2.5: Electromagnetic spectrum.

While this spectrum is rather ample, usually thermal cameras only cover a portion of it, which varies depending on the model and the application for which it is intended. Besides sensing the radiation being emitted from objects in the field of view (FOV), another critical aspect concerning thermal cameras is the ability to measure temperature.

Temperature measurement could easily allow the detection of fire due to the corresponding heat spike generated. However, not all thermal cameras come equipped with radiometric sensors, which adds to the difficulty of the identification task considering the high variability of image context. Later in chapter 3, an extended study is made of the implications this difference between radiometric and nonradiometric cameras has in the data acquired by both.

In general, thermal imaging systems are expensive, and too large and heavy to be used onboard small UAVs, due to payload and autonomy limitations. Recently, several small size cameras, figure 2.6, were brought to market to provide a solution for equipping these vehicles with thermal imaging sensors. That is precisely the case of the model employed in this work, FLIR Vue Pro.

Figure 2.6: On-board thermal cameras.

FLIR Vue Pro

Table 2.2: Summary of FLIR Vue Pro specifications.

FLIR Vue Pro 336 9mm	
Size	63 x 44.4 x 44.4 mm (including lens)
Weight	92.1 - 113.4 g
Sensor Resolution	336 x 256 pixels
Focal Length	9.0 mm
Horizontal FOV	35^o
Vertical FOV	27^o
Spectral Band	7.5 - 13.5 $\mu\mathrm{m}$

As specified in table 2.2, the FLIR Vue Pro records in the 7.5 to 13.5 $\mu\mathrm{m}$ spectral band, meaning it operates in the region denominated as the Far Infrared (FIR) or LWIR, as figure 2.5 indicates.

A key requirement for a fire identification application is that images are available in real-time, so that an algorithm can be developed for online classification. For FLIR Vue Pro, image capture can be triggered either by pairing the camera through bluetooth with a mobile app, or remotely using PWM signals.

Although this camera has MAVLink compatibility for automated image capture and image geotagging, through a port connection that enables access to two camera controls over PWM commands, it is impossible to have access to the images in real-time, only to record them in a microSD memory card. While this limitation is significant, as metadata that could be automatically recorded is lost, a feasible workaround is to trigger the camera via PWM and transmit video over to a ground station. The functions of the PWM commands have to be configured by changing the settings in the FLIR UAS mobile application, and can be programmed to change color palettes, start and stop recording, or trigger the camera zoom from a radio controller.

In fact, using video is a superior solution as it provides a higher degree of flexibility, because it allows a variable timelapse. Fire is a dynamic event that evolves over time and as such the analysis of temporal features is of concern. However, at this exploratory stage it is still uncertain which time interval would be more adequate to yield the best classification possible. So, by recording video instead of single images, the video frames can later be discretized with different sampling frequencies to assess which works best.

Additionally, the quantity of frames that can be extracted from video greatly outnumbers the amount that could be recorded through image capture. The smallest sampling for timelapse available for image capture is a 1 second interval, as can be set in-App for this camera. In contrast, for video recording the frequency is 8.3 Hz, which means that video yields 25 frames every 3 seconds, which is more than 8 times what could be achieved through image capture.

Having described the data acquisition system implemented and presented its principal components, what follows is the explanation of the conditions in which the experimental tests were conducted.

2.2 Experimental Tests

An objective of this work is to assess the behavior of thermal cameras in an array of fire situations. To that end, trials were carried out indoors at Laboratory for Forest Fire Studies (LEIF), figure 2.7, in collaboration with a team from the Association for the Development of Industrial Aerodynamics (ADAI).

Figure 2.7: LEIF in Lousã, Portugal.

For one of the experimental setups tested, figure 2.8, a UAV was equipped with the image acquisition system as previously presented, and short duration flights were performed to record fire ignitions and initial spread.

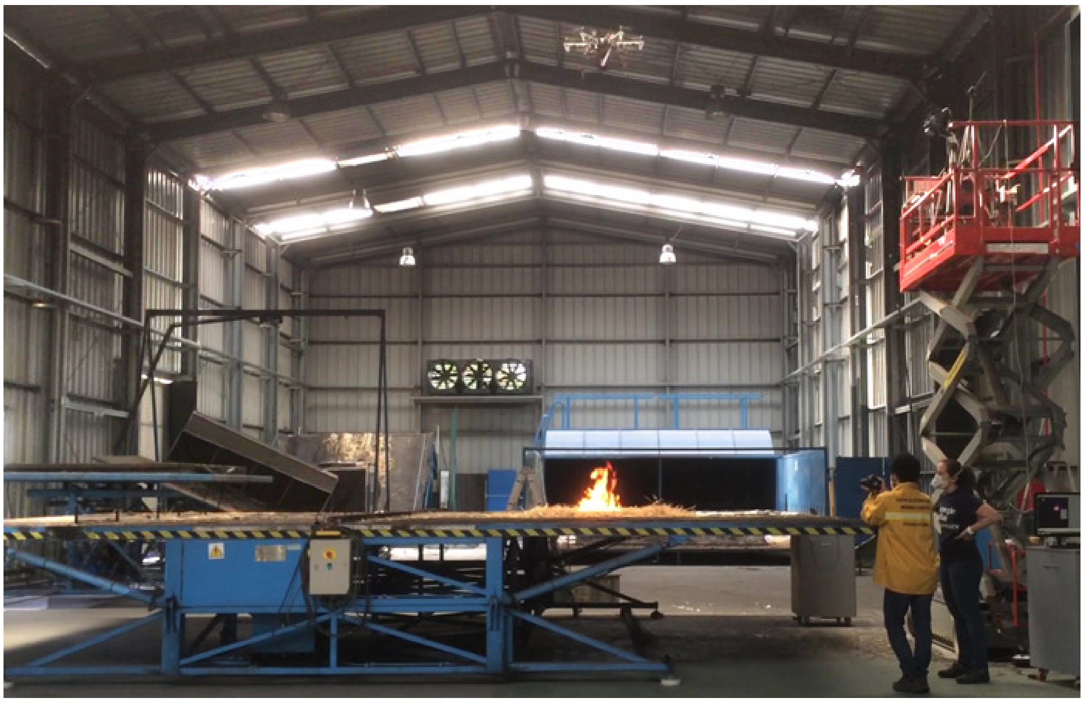


Figure 2.8: Experimental setup with a drone and static platform.

Using a controlled environment in these experiments allowed for the acquisition of video sequences from static and moving platforms, as well as from different types of thermal cameras. In order to have a ground truth for this preliminary study, in addition to the two cameras presented in the previous sections, data was also recorded from an elevated platform using a radiometric thermal camera, FLIR SC660.

The same devices were employed for another type of test related to camping context, in which a tent was burned to study how having obstacles in between the camera and the fire source affects computer vision algorithms. This time around, image capture was performed from a static elevated platform as figure 2.9 shows.

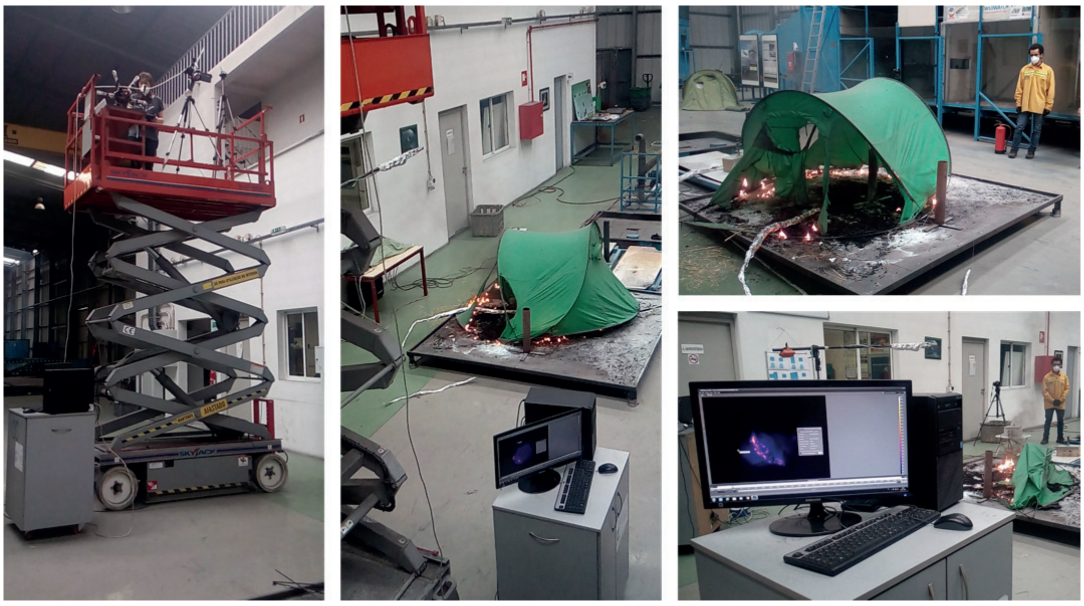
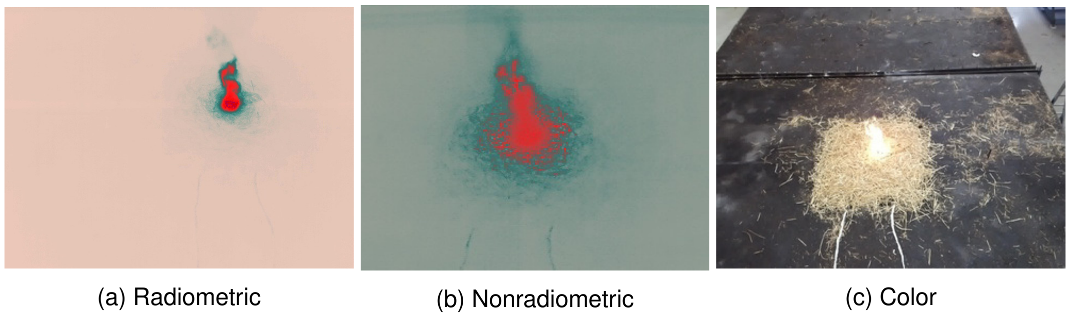


Figure 2.9: Experimental setup for the burning of a tent.

For these indoor experiments images were gathered with three different devices, which resulted in images as the ones illustrated in figure 2.10, for the same instant in time.



(a) Radiometric (b) Nonradiometric (c) Color

Figure 2.10: Images acquired with (a) FLIR SC660, (b) FLIR Vue Pro and (c) Pi Camera V2.

Comparing the images obtained using the three video sources, it is noticeable the difference in type of sensory data, as well as in the field of view, which is due to the fact that each camera has its particular resolution. The image from the FLIR SC660 has a 640x480 spatial resolution, and FLIR Vue Pro with 336x256, has almost half the first resolution diagonal. Because both these cameras have a 4:3 aspect ratio, the color camera was configured to capture video in the same aspect ratio, but with a resolution of 1640x1232. This particular resolution was chosen so that the camera recorded the full sensor area, in order to cover as much area in the FOV of the camera as possible.

Table 2.3 presents a summary of the tests developed, as well as the respective conditions.

Table 2.3: Summary of the experimental tests.

Experimental Tests				Data Acquired		
ID	Description	Conditions	Platform	Color	Thermal Radiometric	T. Nonradiometric
Test 1	Straw Burning	Static	Elevated Platform	✓	✓	✓
Test 2	Tent Burning	Static	Elevated Platform	✓	✓	✓
Test 3	Straw Burning	Moving	Drone	✓	✓	✓
Test 4	Boom Festival	Moving	Balloon	–	–	✓

The use of both radiometric and nonradiometric thermal cameras in the laboratory tests enables the development of an analysis of the sensor response, and understand how image statistics change for the set of experimental scenarios. In the following chapters, these cases will be examined in detail. The focus on color imaging will be addressed further down in chapter 6.

Since further description of these trials is more valuable in context with the approach for each type of data, this is done extensively in each respective chapter. Table 2.4 breaks down the sections where each type data is featured.

Table 2.4: Summary of the analyses applied to the experimental tests.

Chapter Information			Data Acquired		
Chapter	Description	Tests	Color	Thermal Radiometric	T. Nonradiometric
3	Thermal Imaging	1, 4	–	Section 3.3.1	Section 3.3.2
4	Color Segmentation Heuristic	1, 2, 3, 4	–	Section 4.2	Section 4.2
6	Implementation	1, 2, 3	Section 6.4	–	–

Chapter 3

Thermal Imaging

Typically, thermal cameras only produce monochromatic images, unlike color cameras, which make use of 3 color channels, e.g., RGB. That is precisely the case of FLIR Vue Pro, which in the raw format produces a 14-bit monochromatic image in grayscale with 16384 levels of intensity. By employing an array of proprietary camera features, the camera firmware processes the video from the monochromatic raw format to a video encoded in pseudo-color, according to the filter selected. First, the amounts of radiance registered by the sensor are converted to a grayscale image with intensity levels. Only then, the firmware attributes to these levels a scale of colors, according to a specified palette.

At this juncture, this brief introduction serves to explain on a high-level how thermal images are processed. Throughout this chapter, as we explore thermal imaging metadata, we will see this process has more layers of complexity.

In thermal imaging, a variety of color palettes is available to enhance the visual interpretation of the different amounts of radiance captured by the sensors. In this work, the *GrayRed* filter was selected, which helps to draw attention to the hottest objects in a scene, by applying high-contrast color values with a divergent color scheme. The *GrayRed* color palette, which is composed by 120 discrete color samples, can be visualized in figure 3.1.

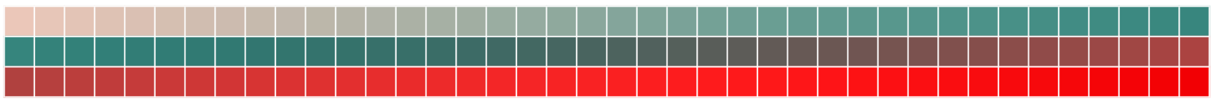


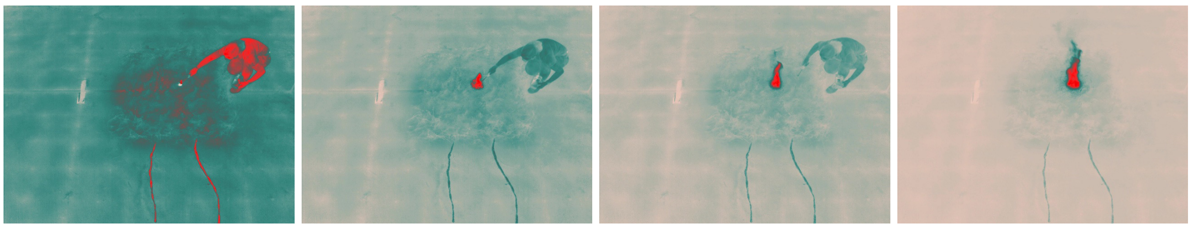
Figure 3.1: *GrayRed* color palette.

While *a priori* it is unclear if employing this color palette specifically will yield any benefit in classification performance, FLIR preset filters will be used as it facilitates the development of future work, providing a basis for comparison of results. For the same reason, the same color palette was applied to the raw data acquired with FLIR SC660, using FLIR ResearchIR software.

3.1 Thermal Imaging Data

As mentioned previously, when cameras are equipped with radiometric sensors, with access to commercial software it is possible to not only have an image where contrast depicts temperature differences, but also have the temperature metadata for each pixel. This section begins by studying an example of radiometric data, and later proceeds to nonradiometric instantiations.

The next example, illustrating a fire ignition, figure 3.2, explores spatial differences in the images first, and later the temperature profile of the image scene is analyzed. Afterwards, the metadata obtained from radiometric and non-radiometric images are compared for different situations.



(a) Frame 80 (b) Frame 86 (c) Frame 121 (d) Frame 167

Figure 3.2: Fire ignition detection with a radiometric thermal camera.

As can be observed in figure 3.2a, right before the ignition, the majority of the frame has green colored pixels, whereas in figure 3.2b, as the fire starts contrast shifts, the objects with temperatures close to the ambient temperature are represented by pixels with a whiter shade.

It is important to notice that the scale that illustrates the differences in temperature adjusts with respect to the object in the scene with the maximum temperature. That can observed by comparing the first and second frames. Whereas, in the first the person is in red, in the second the person is depicted with a dark green. This happens because radiation being emitted by the greatest heat source is measured by the sensor with a much higher intensity than the radiation emitted by the person. While the temperature of the person remained unaltered, the sensor scale of the camera suffered an adjustment. This makes perfect sense, because now the body temperature is much closer to the ambient temperature than to the temperature of the fire. Comparing the first and last frames highlights that as the temperature differences increase the contrast in the image adjusts accordingly. The temperature profile of this example can be observed in figure 3.3.

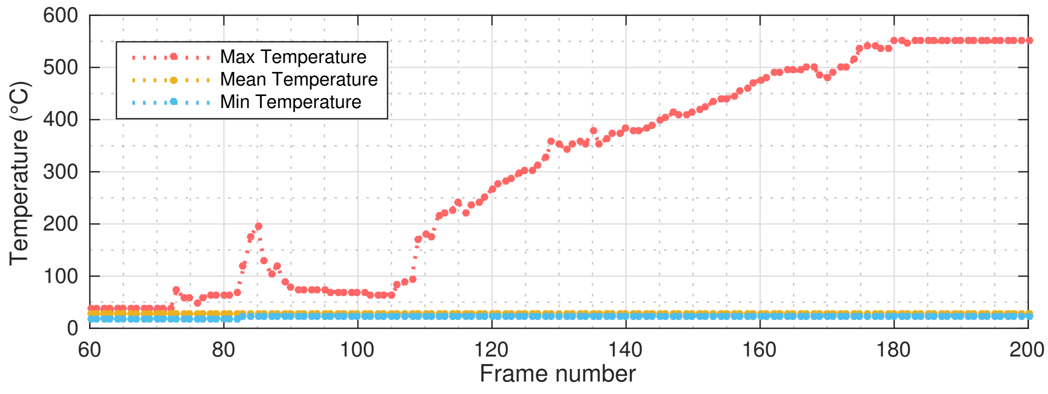


Figure 3.3: Temperature profile for a fire ignition.

As can be seen from the temperature graph in fig. 3.3, the first two frames illustrated in fig. 3.2 do not correspond to a fire ignition with its full extent, since the temperature decreased after the initial spike. Yet, these are enough to highlight that a small source of very high temperatures is sufficient for the sensor to adjust the color of the surroundings in the scene. This demonstrates that radiometric cameras have high sensitivity, thus being adequate to provide an accurate ground truth at this stage.

Conversely, for nonradiometric thermal images the metadata is limited to the raw digital signals outputted from the sensor analog/digital converter. These raw values represent the levels of the sensor bit resolution, that is a characteristic of each camera.

In this work, two different thermal cameras were used, a radiometric model, FLIR SC660, and a nonradiometric, the FLIR Vue Pro, presented in detail in section 2.1.3. These have distinct spatial, bit and temporal resolution, which are specified in table 3.1.

Table 3.1: Sensor resolution.

Camera	FLIR SC660	FLIR Vue Pro 336 9mm
Spatial Resolution	640 x 480	336 x 256
Bit Resolution	16-bit	14-bit
Temporal Resolution	15 Hz	8.3 Hz

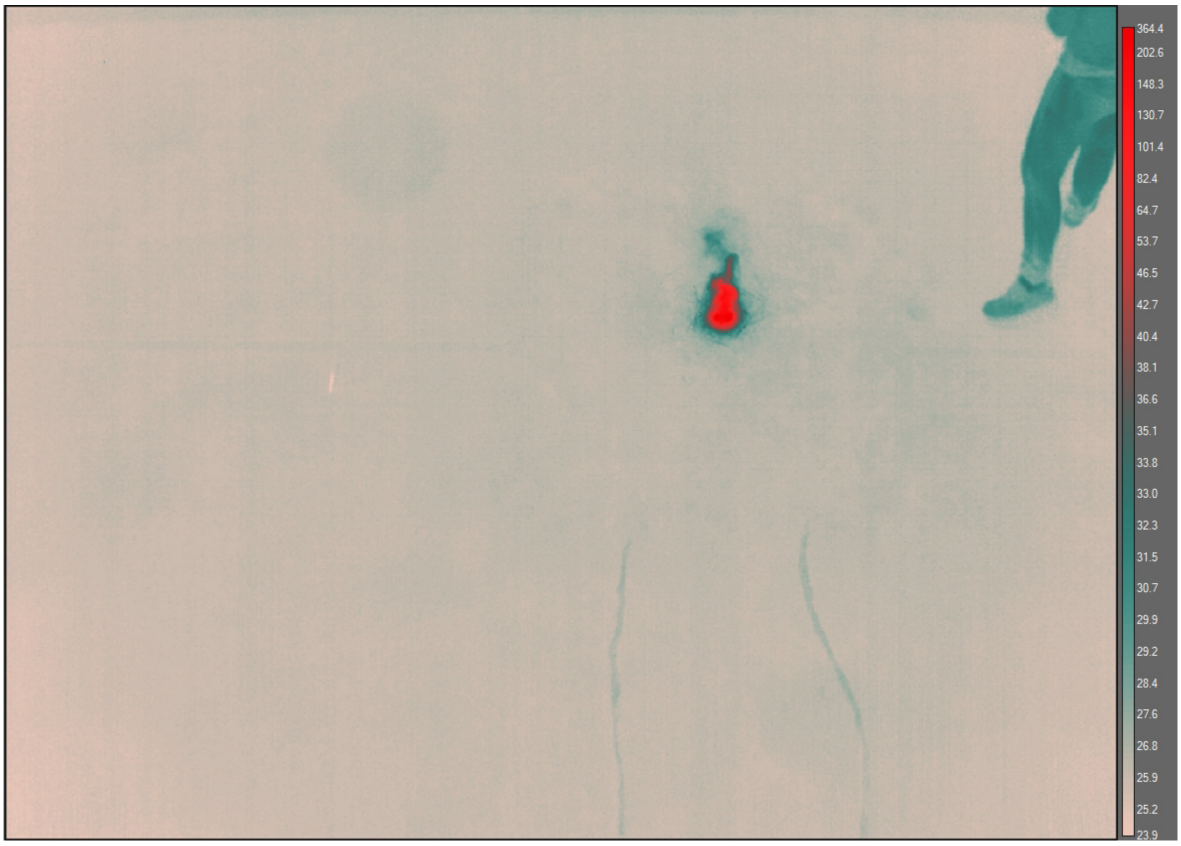
This chapter deals exclusively with raw data for both cases separately, so spatial and temporal resolution are not a concern at this time. In the following chapters, however, the data from both cameras are presented side by side, and the difference in temporal resolution presents a difficulty for the comparison of results of videos from both cameras. To avoid the issue concerning temporal resolution the videos of FLIR Vue Pro were oversampled and converted to 15 Hz using FFMPEG so that the frames from each camera match. Regarding the spatial resolution, images were left with the original size since it highlights aspects related to data variability when dealing with different thermal imaging sensors.

Using the FLIR ResearchIR software, the metadata encoded in the images can be visualized interactively, and we have access to the scale of temperatures for the radiometric case. These temperatures vary in each frame as the scale adjusts to the maximum temperature in the scene. As an example, figure 3.4a depicts a fire situation where the temperature scale covers the full range of the color palette, with pixels of red, green and gray tones. Similarly, for the non-radiometric case, figure 3.4b, the software applies the scale according to the intensity levels captured, but there is no temperature assignment. Notice, however, this instance is in a laboratory environment and the areas depicted in red are not fire, but 250 W lamps[1] whose temperature can exceed 300^{o}C.

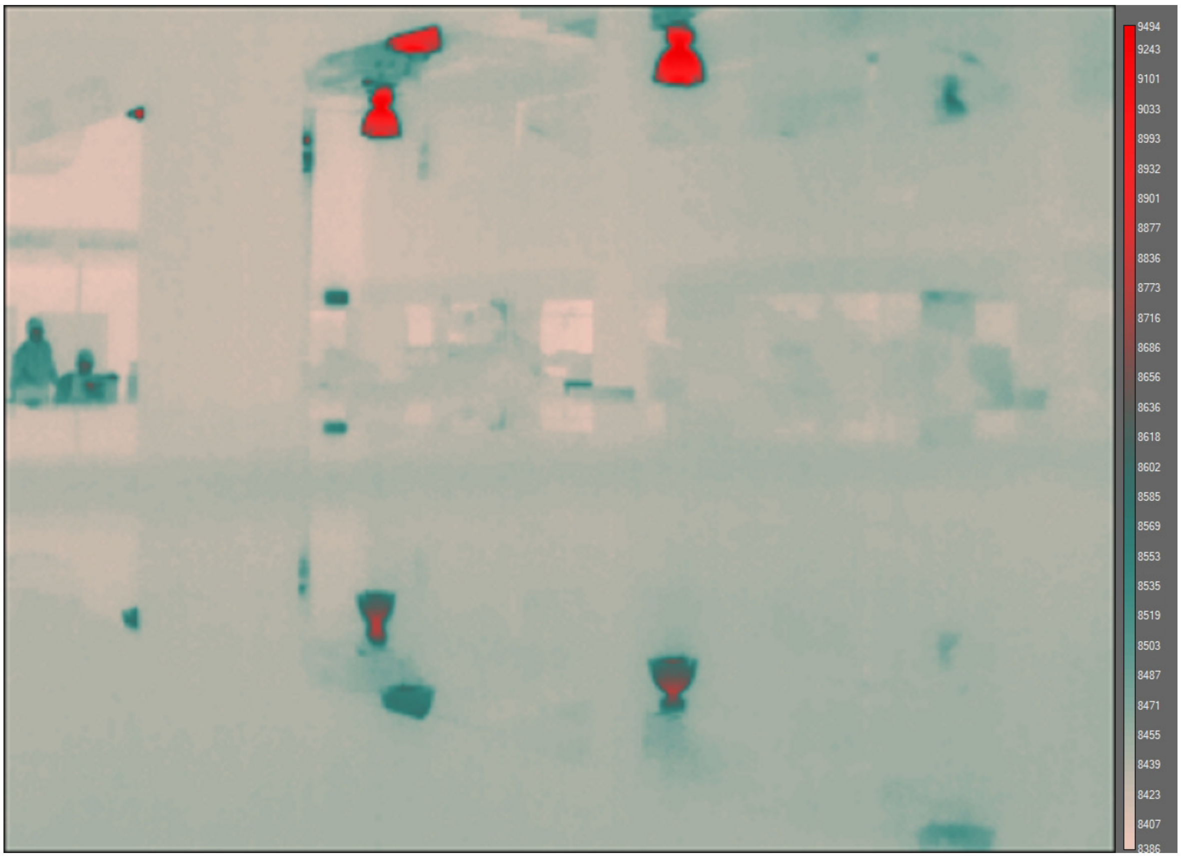
For both cases, the intensity level is given by a digital number assigned by the sensor analog to digital converter. The difference is that for radiometric data these raw signals are converted to a temperature value according to the camera calibration. Nonetheless, these measurements should be corrected *a posteriori*, since the setup for experiments is not always the same. Consequently, the encoded radiometric data will not be accurate if the measurements are not properly adjusted.

Since the color scale adapts as temperatures in the scene change, the following sections will center attention on understanding how the sensor reacts to what is actually being measured.

[1]Lamp reference: Osram Powerstar HQI-E 250W/N/SI E40.



(a) Radiometric data



(b) Non-radiometric data

Figure 3.4: Comparison of thermal imaging data.

3.2 Sensor Measurements

One of the main difficulties of interpreting thermal imaging data is the constant color adjustment of the image. In order to get a better sense of what is the output of the sensor prior to RGB encoding, it is convenient to first analyze how the raw digital signals and the respective temperature measurements evolve over time. Figure 3.5 presents this comparison for a fire ignition captured from an elevated platform using a radiometric camera.

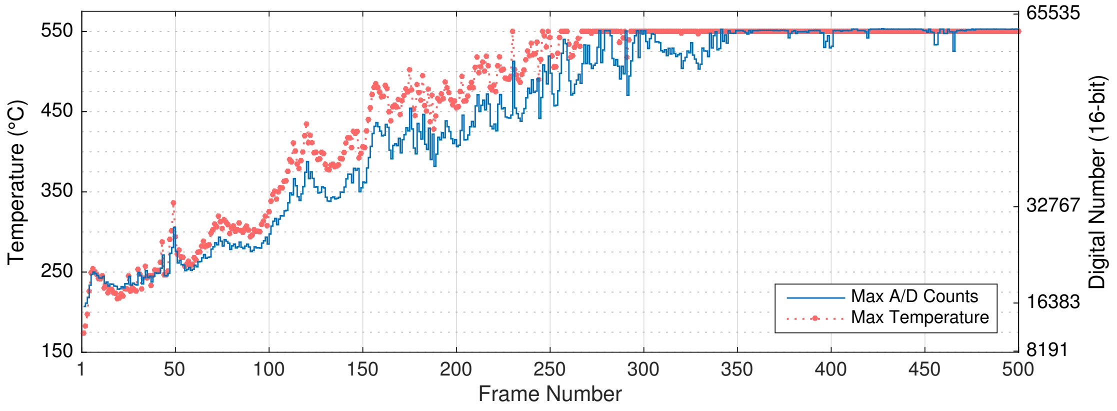


Figure 3.5: Temperature versus A/D Counts.

The most immediate observation from the graph is that the change in maximum temperature is highly correlated to the increase in the maximum value of A/D counts for each frame. This gives the indication that we should investigate deeper how the software computes the temperature data. Furthermore, from the graph in figure 3.5, it is interesting to note that around frame 350, the sensor output of maximum raw signals is close to the limit of the sensor resolution, 2^{16} = 65536 levels, and this corresponds to a maximum temperature of 550ºC. Although the official datasheets of FLIR SC660 mention an option of extended measurement range up to 2000ºC, in this instance the camera is clearly configured for a lower temperature interval.

Granted that there is a strong relationship between the sensor digital output and the temperature values, if we were to know the parameters responsible for this conversion, it would be possible to apply the same relation to the raw digital signals from the non-radiometric camera. Hence, it makes sense to perform this analysis, for data from both cameras, at the same time. Making this study in parallel allows for comparison with the ground truth data which serves as validation for this approach.

In order to clarify what is the actual working temperature range, we will resort to the metadata encoded in the image files for both cameras. In the case of the radiometric camera, these are SEQ files, that include multiple sequential frames with the raw digital values. Conversely, for the nonradiometric camera, both TIFF and JPEG files are used. The first can be of single or multiple frames, and the latter include both the RGB encoded image as well as the raw TIFF file. The metadata from these files can be extracted using Exiftool, an application capable of handling a variety of metadata formats.

By using Exiftool, it was possible to discover the temperature range of measurements for both cameras, as listed in table 3.2. These camera parameters are characteristic of each particular sensor.

Table 3.2: Camera Temperature Limits.

Camera	FLIR SC660	FLIR Vue Pro 336 9mm
Temperature Range Max (°C)	500	135
Temperature Range Min (°C)	0	-25
Temperature Max Clip (°C)	550	150
Temperature Min Clip (°C)	-20	-60
Temperature Max Warn (°C)	500	135
Temperature Min Warn (°C)	0	-25
Temperature Max Saturated (°C)	550	150
Temperature Min Saturated (°C)	-60	-60

Observing the temperature intervals of both cameras it is simple to verify that they have quite distinct sensing capabilities. This new information listed above is not provided in the official datasheets, but is extremely important. In what regards FLIR Vue Pro, the fact that the sensor saturates at 150°C can potentially be a limitation for a fire identification application. Considering that fire reaches much higher temperatures, a detection algorithm solely based on temperature would most likely be prone to false alarms.

Calibration Parameters

Also with Exiftool it was possible to uncover the full list of camera parameters for both cameras, which includes several constants related to atmospheric radiation transmittance. Considering that one may not be familiar with using Exiftool and command line instructions, the camera calibration parameters are listed in table 3.3, since they can be useful for future reference.

Table 3.3: Camera Parameters.

Camera	FLIR SC660	FLIR Vue Pro 336 9mm
Emissivity	0.95	0.94
Object Distance (m)	1.0	50.0
Reflected Apparent Temperature (RAT) (°C)	20	22
Atmospheric Temperature (°C)	20	22
IR Window Temperature (IRWT) (°C)	20	22
IR Window Transmission	1.0	1.0
Relative Humidity (%)	50.0	45.0
Planck R1	17647.318	17096.453
Plank B	1484.7	1428
Planck F	1.9	1.0
Atm. Trans. Apha 1	0.006569	0.006569
Atm. Trans. Apha 2	0.012620	0.012620
Atm. Trans. Beta 1	-0.002276	-0.002276
Atm. Trans. Beta 2	-006670	-006670
Atm. Trans. X	1.9	1.9
Planck O	-7840	-342
Plank R2	0.082014173	0.046642166
Peak Spectral Sensitivity (μm)	9.7	10.1
Raw Format	16-bit	14-bit

While these parameters give some insight in what are the parcels at stake for the temperature conversion, relating the names of these constants to the variables they represent is not immediate. Whereas the units of some parameters are easy to infer, like the temperature expressed in degrees Celsius, most of them are not clear at all. Table 3.3 is purposely divided into 4 sections, as the items in each relate to different parts of the computation. The constants are displayed in the order they are encoded in the metadata, but this is not necessarily the sequence in which they are used.

3.3 Adaptive Color Scale

As mentioned previously, understanding how the camera adapts the color scale is one of the principal difficulties in interpreting thermal imaging data, which is essential to devise a robust computer vision algorithm. For FLIR Vue Pro the adaptive color assignment depends on an array of camera features such as Digital Detail Enhancement (DDE), Active Contrast Enhancement (ACE) and Smart Scene Optimization (SSO). These algorithms tune the color assignment and can be adjusted manually using FLIR UAS mobile app. More details about these controls can be found on the official documentation [28], but for this work default settings were used.

From the analysis of the metadata of the image and video files, two exif tags were identified that have paramount influence in adapting the color scale, the ”Raw Value Median” and ”Raw Value Range”.

The median and range values are encoded separately in the metadata of the files, but do not always correspond to the values that would be calculated from the raw frames. The reason for this is probably some proprietary in-camera processing to deal with noise and other type of outliers. For FLIR SC660 the calculated values do not match the encoded values in the files, whereas for FLIR Vue Pro both match perfectly for every image.

With the median and raw values from the metadata tags, the maximum and minimum values of the color scale are computed as follows:

$$\text{Raw Value Max} = \text{Raw Value Median} + \frac{\text{Raw Value Range}}{2} \tag{3.1}$$

$$\text{Raw Value Min} = \text{Raw Value Median} - \frac{\text{Raw Value Range}}{2} \tag{3.2}$$

Thermal cameras are calibrated using black-body emitters, in a slow process to generate calibration curves, with successive small increments in temperature [29]. Instead, in this section, two examples will be presented for the radiometric and non-radiometric cameras used in this work, in operation conditions. The objective is to investigate further the response of the sensor, and the influence of these parameters.

The following examples focus on real situations pertaining to fire identification in camping parks. First, the next case demonstrates the sensor response in a fire situation with extreme temperatures, for the radiometric camera. The second case, in turn, illustrates the sensor response retrieved from nonradiometric data acquired with a balloon at high altitude prior to the development of this work.

3.3.1 Controlled Fire Example

For this example, data were acquired using a radiometric camera, FLIR SC660, on an elevated platform, as depicted earlier in figure 2.8. The purpose of this example is to analyze how a fire event is captured on a sensory level, and understand the way the color scale adapts in such conditions. To that end, figure 3.6 illustrates the evolution of the raw values calculated internally by the camera firmware, which were read from the metadata embedded in the files.

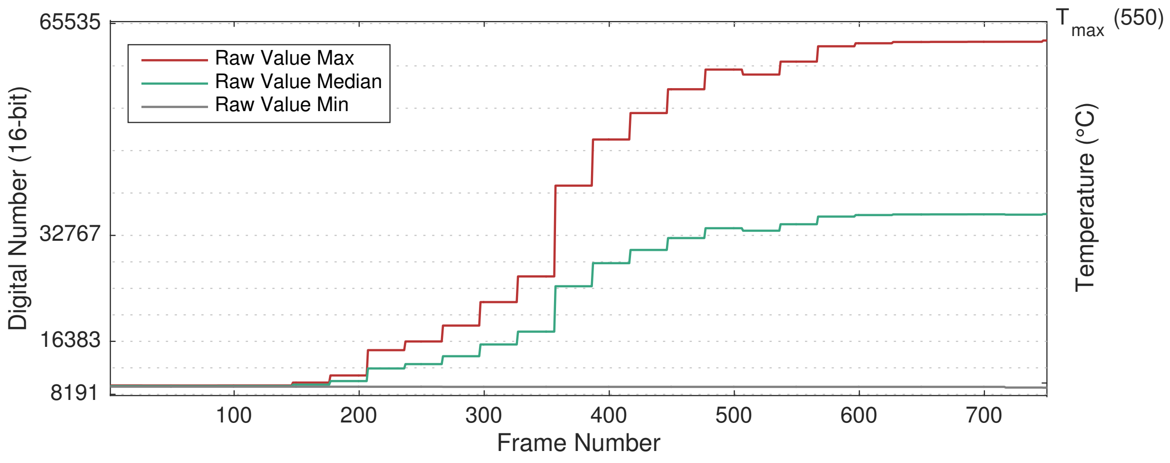
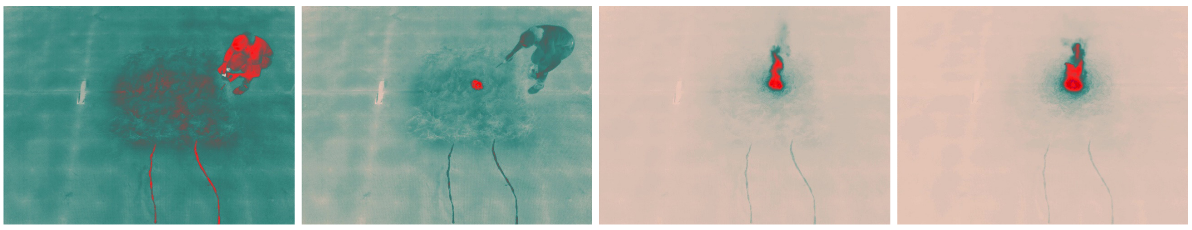


Figure 3.6: Evolution of the range and median raw values for a controlled fire.

On the one hand, this example reveals the camera has a processing level that does not update the scale in every frame but rather in steps of 30 frames. Taking into account this device works at 15 Hz, corresponding to 15 frames per second, the color scale is updated once every 2 seconds.

On the other hand, this graph demonstrates the stretching of the color palette as maximum raw values increase, which attests to the fact that the filter adjusts the scale with respect to the object in the scene with the maximum temperature. Although the color assignment is not a simple linear assignment of values from the color palette, the amplitude of the range for extreme differences of temperature justifies why the green pixels disappear in these conditions. To aid this conclusion, figure 3.7 depicts selected frames from this sequence.



(a) Frame 1 (b) Frame 100 (c) Frame 200 (d) Frame 300

Figure 3.7: Fire ignition detection with a radiometric thermal camera.

The sequence of frames selected is evidence that for a fire situation the mid-range band of the color scale disappears rapidly, which can be verified in the difference between frames 100 and 200 in fig. 3.7. Considering the full extent of the sensor bit resolution, despite this change occurring at a low intensity level, the maximum temperature of the corresponding frames are already above 200°C (see fig. 3.5).

3.3.2 Boom Festival Example

Prior to the development of this work, under the scope of project Firecamp 2, data were acquired at Boom Festival, in Idanha-a-Nova, Portugal. For these tests nonradiometric data were recorded outdoors with resource to a tethered balloon at a significantly higher altitude than the previous examples.

The fact that for these tests thermal data were only collected using FLIR Vue Pro has some drawbacks. On the one hand, the lack of ground truth for this example imposes a considerable difficulty in the interpretation of the images. On the other hand, the characteristics of this moving aerial platform, and the conditions in which these tests were performed are real operation conditions for this imaging device. As introduced in table 3.3, the calibration parameters of this camera take into account atmospheric transmittance and air humidity which are not negligible at higher altitudes.

The tests performed encompass the aerial surveying of the Boom Festival venue, from which two distinct situations were identified. The first, depicted in figure 3.8, does not register strong heat sources in the camera FOV, thus the raw range does not expand.

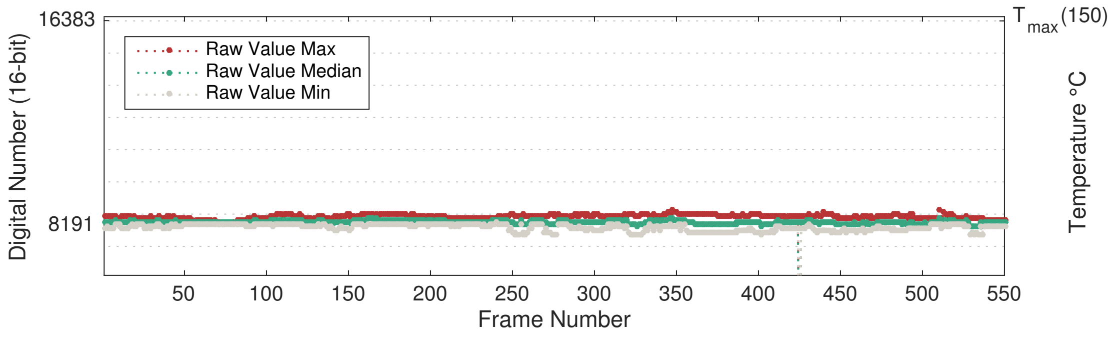


Figure 3.8: Raw measured with Flir Vue Pro from high altitude with a 5 second timelapse.

Although not much can be inferred from this profile, it provides a baseline for comparison with other situations, such as the one is illustrated in figure 3.9.

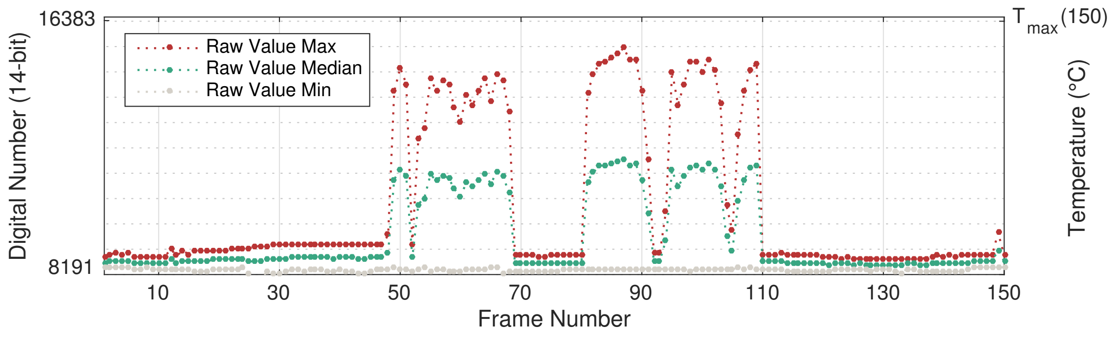


Figure 3.9: Raw measured with FLIR Vue Pro for community kitchen area.

For this instance the sensor registered several spikes, with the raw signals assuming values much higher than the defined baseline. By inspecting the corresponding sequence of frames, presented in figure 3.10, what stands out is that the spikes are triggered when the same area enters the FOV of the camera. With knowledge of the venue premisses, it is possible to identify this place as the community kitchen area, which is equipped with rocket stoves. This is an area where fire detection is to be expected, in addition to other heat sources such as objects at elevated temperatures.

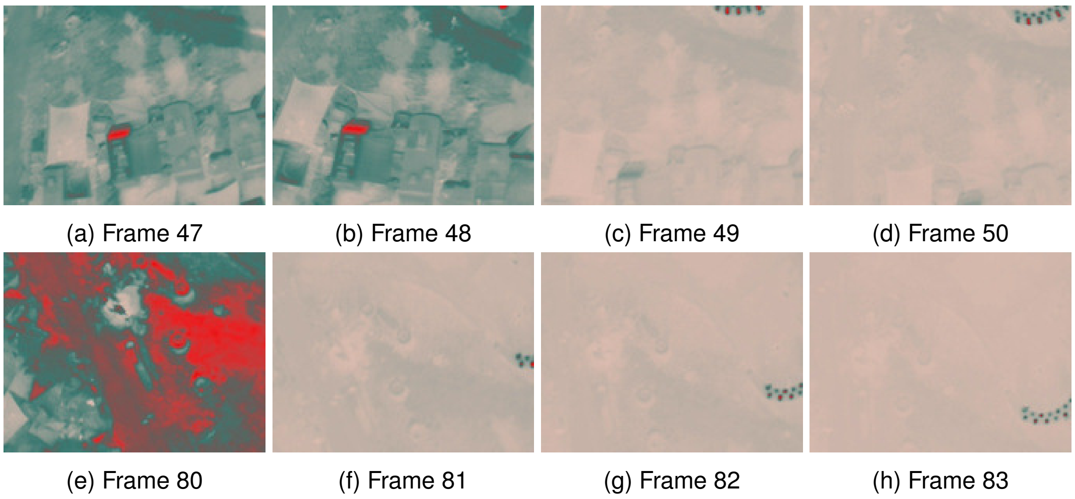

(a) Frame 47 (b) Frame 48 (c) Frame 49 (d) Frame 50
(e) Frame 80 (f) Frame 81 (g) Frame 82 (h) Frame 83

Figure 3.10: Heat detection with a nonradiometric thermal camera.

Regarding the color scale, this example, once again, confirms that when the range of temperatures in the scene expands, the percentage of green in the image reduces considerably. However, considering this camera was programmed to take photographs with a 5 second time-lapse, it is unknown the time the firmware takes to adjust the color scale. Additionally, taking into account the altitude at which the images were captured, this result proves that the color scale adapts even when hot objects have low spatial resolutions. Moreover, this example highlights the requirement of having a ground truth for a fire detection computer vision algorithm, since when the frames become whiter, the objects in the scene are almost impossible to discern.

3.4 Discussion

This initial analysis illustrates to some extent the principle behind how these devices work. However, as this study continues in the following chapter, we will see that this alone is still a very superficial understanding of the capabilities of this type of sensors. Although resorting to the raw digital signal or an actual temperature measurement gives an absolute reference, this approach neglects the content of the images which are rich of relevant information.

Additionally, having identified that the sensor of FLIR Vue Pro saturates at 150°C poses a red flag in what concerns fire detection. The example given from Boom Festival, serves as an indication that a low saturation level of the sensor would lead an algorithm to trigger a substantial amount of false alarms following this approach. A possible solution for this problem would require prior geofencing of low-risk areas such as the community kitchen area. However, at the other end of the stick, if a fire incident was to start precisely in that area the algorithm would be blind to it. Notwithstanding, from a safety standpoint, the aim should not be to find a universal solution, but rather to have redundant sensors to ensure most risk situations are covered. In that sense, the monitoring of this particular variable could be of great value, if included in a broader surveillance system.

Hence, to further explore the potential of thermal imaging data, the analysis will continue in the following chapter, with a color segmentation approach.

Chapter 4

Color Segmentation Heuristic

From the literature review, a common computer vision approach to fire detection is through color segmentation [13, 14, 15]. Additionally, from what was just observed in the analysis of the ground truth images, a color segmentation heuristic presents itself as an obvious approach.

Before this work, under the scope of Project Firecamp 2, a preliminary analysis was performed using a segmentation heuristic to attempt to detect fire presence in a real-world setting. In part, this effort focused on red segmentation in the RGB and HSV color spaces, but had some limitations mainly due to exploring only a single feature in image data. Since the hottest objects are depicted in red, and the objective is to detect fire, this assumption seemed reasonable. Now, however, with access to more data as well as radiometric information, this approach will be extended.

4.1 Color Segmentation

Segmentation is a computer vision process which consists of dividing an image into regions according to a particular image property, e.g., gray level, color, texture, depth or motion. Taking into account the nature of thermal images, color segmentation will partition the image into mutually exclusive regions in function of the color of each pixel. Therefore, since the color assignment depends on the relative temperature between all the pixels in the scene, it is in fact segmentation by temperature.

While, even for a non-expert in thermal imaging, this interpretation might appear to be simple, translating the human vision perception to a robust computer vision algorithm is not always as straightforward. With the objective of developing a segmentation heuristic, the study of the example described in section 3.1 will now be continued.

Considering that thermal images in pseudo-color are used to highlight differences in temperature, a segmentation heuristic should take into account the color bands used for such purpose. Recalling the images in figure 3.4, these colors are red, green and gray, as assigned by the *GrayRed* filter. The access to the color assignment of the filter is essential to devise the segmentation heuristic, without it we would be picking color ranges blindly.

To illustrate how the division of the color palette was made, figure 4.1 shows the extension of each partition of the scale.

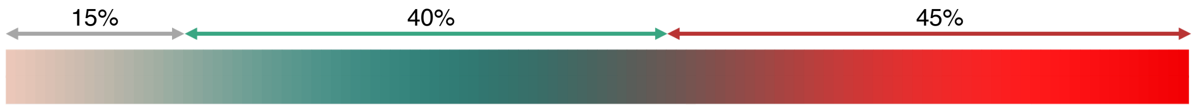


Figure 4.1: Division in color segmentation classes.

The complete scale of 120 discrete color values, which depicts increasing temperature from gray to red, was divided into three segments. First, the gray segment includes 18 color levels, which represent 15% of the full scale. The mid-range is composed of 48 colors of green tones, which correspond to 40% of the scale. The remainder is the red part, that comprises the last 45% of the color palette, and is defined by 54 color levels. Therefore, from the colors presented in figure 3.1, the segmentation bands are defined for the three classes as specified in table 4.1.

Table 4.1: Segmentation thresholds.

Gamma	Gray			Green			Red		
Channels	R	G	B	R	G	B	R	G	B
Upper limit	253	199	185	143	169	157	255	73	71
Lower limit	149	171	160	98	90	86	103	89	85

Although the colors in table 4.1 are represented in the RGB color space, the division in these three classes helps to convey how the color distribution varies over time. In opposition, if this analysis were to be done in terms of absolute values of each color channel (Red, Green, and Blue), the result would be less intuitive and of complex interpretation.

In order to extend the analysis done previously, the radiometric data is juxtaposed with the color segmentation results as depicted in figure 4.2. There are several regions in the graphs that are of particular interest, so this analysis will start with an overall view before getting into more specific insights.

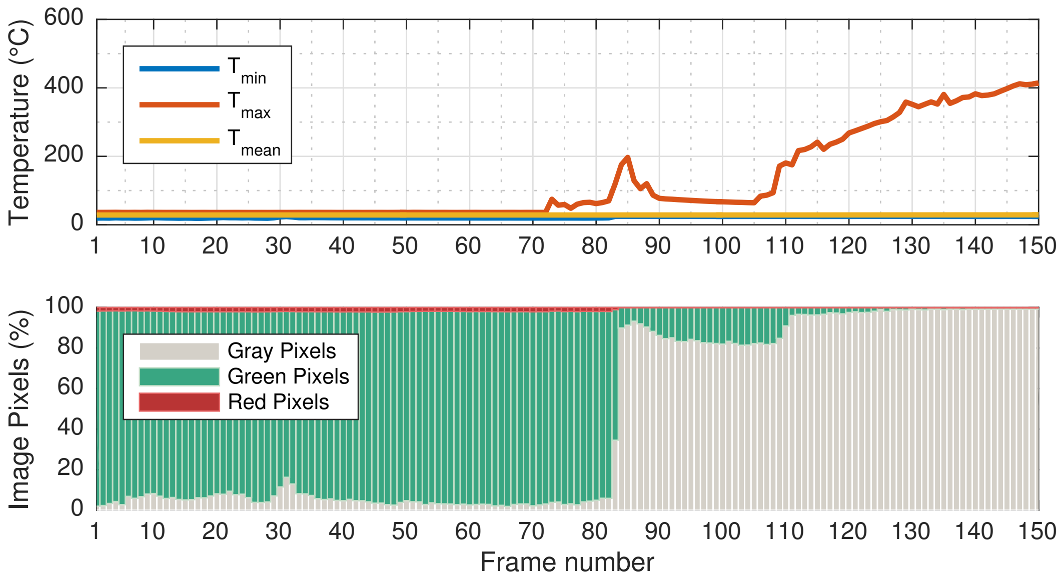


Figure 4.2: Fire detection with radiometric camera for Test 1.

First, in the region up until the ignition, around frame 70, green represents the largest percentage of the image pixels, with white and red representing a small amount for this case. Then, around the mid 80's there is a considerable shift, where white pixels become predominant instead of the green pixels. This occurs due to the change in maximum temperature and demonstrates that the most relevant features in the image are the percentage of green and white pixels.

Contrary to what was previously thought, this reveals that the hottest points are not the most relevant after all, and this straightforward example of a simple fire ignition further shows that the area of red pixels is by far the one with the least representation. Hence, by virtue of the analysis now made, it becomes evident that monitoring the variation in red percentage was a misconception, since it is clear that the most significant alterations occur for the green and white percentages of the image.

Conversely, focusing on the temperature profile, around frame 73, the small bursts in temperature are justified by the ignition of the lighter, in testing prior to the beginning of the fire, and shows how sensitive the measurements with the radiometric camera are. At the same time, in the pixel graph, for the same frames, it is clear that the camera does not respond immediately to this change. Only when the temperature differences become more extreme, after frame 80, a steeper alteration in the image color distribution takes place. The reason for this might be the fact that the camera only adjusts the range of the color scale every 30 frames, as was identified in section 3.3.1.

In turn, examining the ending section of the graph, around frame 110, it can be observed that there is a slight increase in the percentage of gray pixels. Thus, confirming the tendency that as the maximum temperature increases and the difference between maximum and minimum temperatures is greater, the image takes a lighter tone.

Up until now, the frames from 80 until 110 were purposely left out of the analysis, because they require a more comprehensive understanding of what happened and that is not visible through the previous images. In this test scenario there was first an initial ignition that did not start the fire completely, thus requiring a second ignition. That is the reason why there is an initial peak in temperature at frame 86, but the gradual increase in maximum temperature only occurs from frame 110 onwards.

Although at first glance this profile might not seem ideal for this analysis, it actually highlights perfectly an essential consideration in the development of an algorithm for fire detection. This case is an example of what could potentially represent a false alarm, and should, or not, be picked up by the algorithm depending on how conservative the threshold for an alarm is.

Nevertheless selecting an adequate threshold requires a comprehensive set of target situations. As such, for the success of this type of approach, it is imperative to have a clear definition of what are the risk situations to be targeted first. As a starting point, the data relative to the experimental trials will be analyzed.

At this preliminary stage, the intent is to explore the data collected and ascertain the limitations associated with thermal imaging. As such, there will be no further feature transformation at this point, since it would be detrimental to the objective of understanding the capabilities of different sensors. To that end, the following sections will analyze in detail the array of situations tested.

4.2 Validation Examples

The experimental trials cover three distinct scenarios of fire risk situations in camping parks. These situations encompass burning of forest fuels like straw, but also include highly combustible materials characteristic to the camping context. The first scenario consists of a controlled ignition of straw, and the second corresponds to the burning of a camping tent, both in a laboratory environment. The last is a real-world setting in a camping park at Boom Festival, though no fire incidents have occurred.

For the laboratory tests, distinct setups were used to study different aspects related to the performance of computer vision algorithms. This section will focus on presenting the objectives of each of the tests and describe the conditions in which they were carried out. Additionally, some initial comparisons are made from data acquired with both cameras. Subsequently, in section 4.3, the results from the segmentation heuristic will be presented along with the discussion of the outcomes of this approach.

4.2.1 Straw Examples

To assess the advantages and limitations of thermal imaging it makes sense to start from simpler cases and work up to more complex ones. Following that reasoning, for the controlled ignition of straw, two types of tests were conducted. For the first case, videos were recorded with both radiometric and non-radiometric cameras from a static elevated platform. Some images corresponding to this test are presented in figures 4.3 and 4.4, respectively.

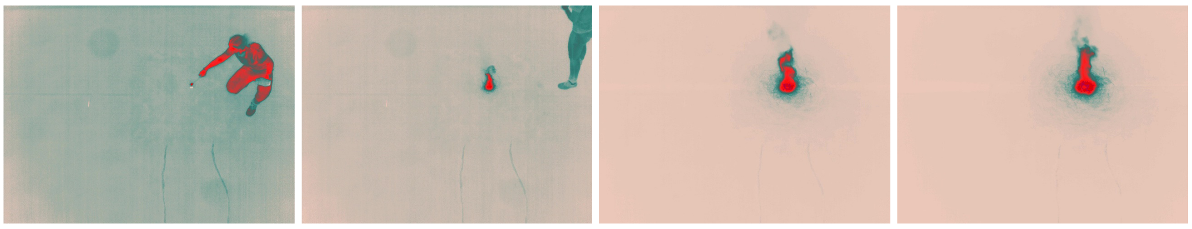

(a) Ignition point (b) 2.5 seconds later (c) 15 seconds later (d) 25 seconds later

Figure 4.3: Fire detection with Flir SC660 (radiometric).

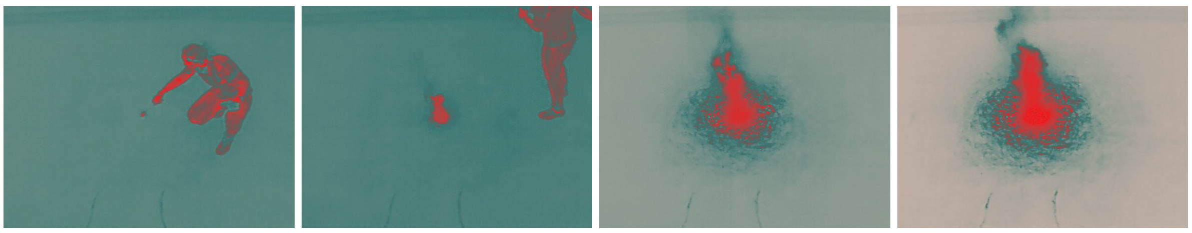
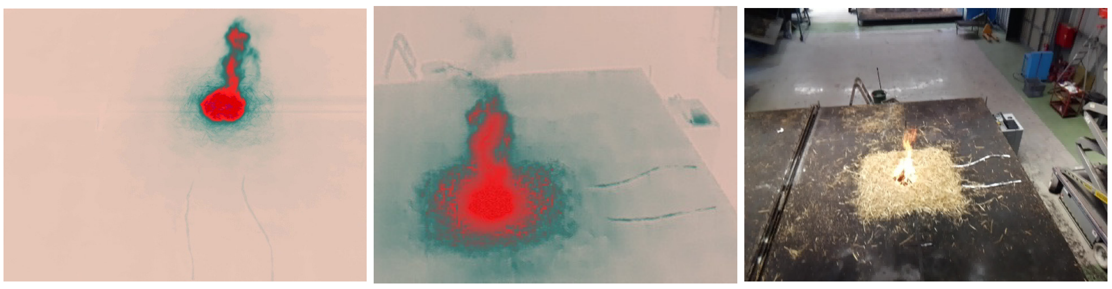

(a) Ignition point (b) 2.5 seconds later (c) 15 seconds later (d) 25 seconds later

Figure 4.4: Fire detection with Flir Vue Pro (nonradiometric).

By comparing the images in figures 4.3 and 4.4, it is possible to observe that the distinct FOV will affect the image statistics. Moreover, a noticeable difference is that the color adjustment after ignition occurs at a much faster pace for the radiometric sequence in figure 4.3. The reason for this is that for the radiometric images, the color assignment is done externally to the camera using Flir ResearchIR software, whereas the non-radiometric pseudo-color images are encoded in-camera.

A second test again considers the burning of straw, with the radiometric camera positioned in an elevated platform. The nonradiometric image acquisition was, in this case, performed from a moving platform, using the UAV presented earlier in chapter 2. To illustrate the new vantage point, figure 4.5 juxtaposes images acquired from the radiometric camera on the platform to a nonradiometric and a color image acquired with the aerial system mentioned.



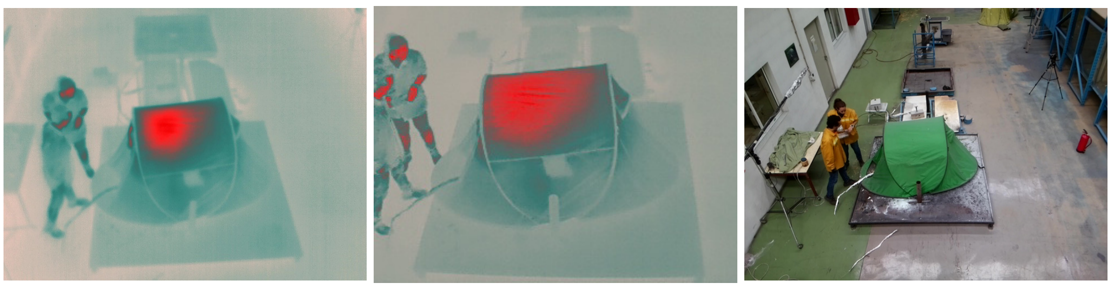
(a) Radiometric image (b) Nonradiometric Image (c) Color Image

Figure 4.5: Perspectives with 3 different cameras.

Both cases presented so far comprise the same type of test, and data were recorded with the three devices, with the only difference being the point of view for the second case. These instances account for situations where the fire is visible to the naked-eye, thus could, in theory, be detected using color images also. Notwithstanding, one of the main advantages of using thermal imaging is the ability to see beyond the visible spectrum, thus being capable of detecting sources of thermal radiation to which there is no direct view. The following example aims to harness this advantage in another set of conditions.

4.2.2 Tent Example

In a camping environment, fires can originate from inside tents which means it is important to study these cases due to their elevated propagation potential. An important aspect of this type of instances is that the fire sources are obstructed by a somewhat opaque material. To evaluate this situation, a test was performed under static conditions on an elevated platform, again using the radiometric, nonradiometric and color cameras. Akin to the samples given above for the straw tests, figure 4.6 depicts the different types of data collected, now for the case of the tent test.

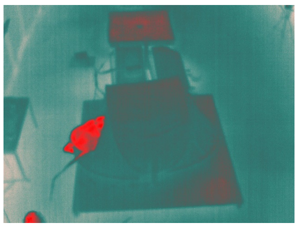


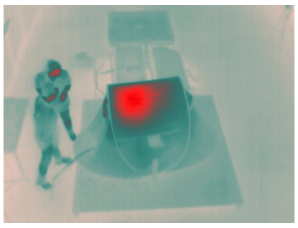
(a) Radiometric image (b) Nonradiometric Image (c) Color Image

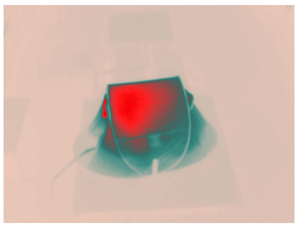
Figure 4.6: Image data from the burning of camping tent.

The images presented in figure 4.6 illustrate a point in time where there is an actual fire inside the tent, but is not visible to the naked-eye as can be observed in figure 4.6c.

In order to study this case, images from both thermal cameras are presented in figures 4.7 and 4.8, respectively. These are complemented with the corresponding sequence captured with color camera, in figure 4.9.


(a) Ignition point


(b) 15 seconds later


(c) 40 seconds later

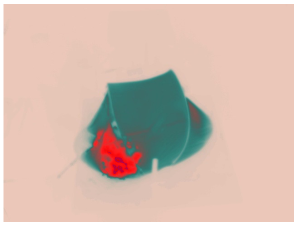
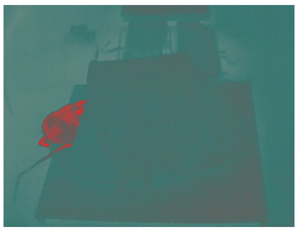
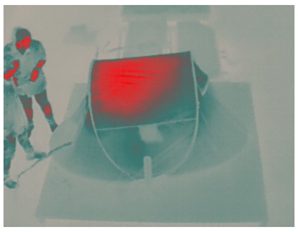
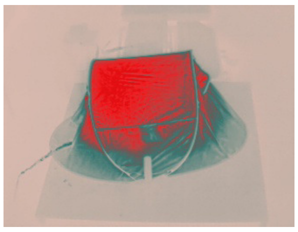

(d) 4 minutes later

Figure 4.7: Tent burning captured by Flir SC660 (radiometric).


(a) Ignition point


(b) 15 seconds later


(c) 40 seconds later

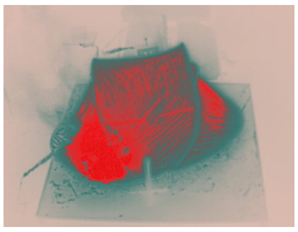
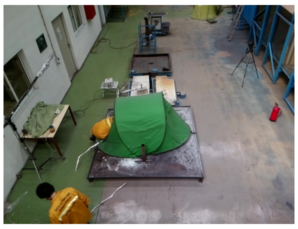
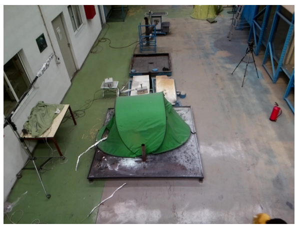

(d) 4 minutes later

Figure 4.8: Tent burning captured by Flir Vue Pro (nonradiometric).


(a) Ignition point

(b) 15 seconds later


(c) 40 seconds later

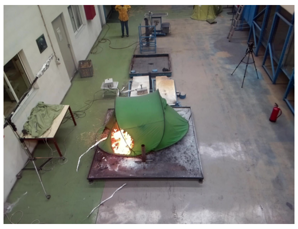

(d) 4 minutes later

Figure 4.9: Tent burning captured by RPi Camera Module.

Comparing with the previous tests, now, at an early stage, the area in red does not depict the actual fire, but the heat built-up inside the tent. Observing the sequences in figures 4.7 and 4.8, it takes a longer time for the camera to adjust the color scale. This extended timespan is due to the fact that a semitransparent object is blocking part of the radiation.

From the color image sequence, in figure 4.9, it is difficult to ascertain if there is a fire occurring until the last frame when the exterior part of the tent is already being consumed, and there is direct view of the fire.

4.2.3 Boom Festival Example

This example concerns the trials from Boom Festival that were already introduced in a previous chapter, specifically in section 3.3.2. In that section, some samples of the tests were already illustrated, and for that reason are not repeated this time around. As discussed earlier, images were captured only with FLIR Vue Pro, so there is no ground truth for this case.

Nevertheless, the analysis done for the previous examples will be performed for the nonradiometric data. The interest in the study of this case is to assess how a segmentation algorithm would cope with the movement of the acquisition system. This is of particular concern because, when capturing images from aerial platforms, perturbations due to wind have a considerable effect on the acquired images. Additionally, the effects of the lack of stabilization of the camera are amplified by its restricted FOV.

4.3 Results

This section explores the results obtained for the different scenarios. To that end, the results are analyzed with two distinct approaches. The first addresses how the levels of each segmentation class vary over time, to clarify the RGB encoding done by the cameras. In turn, the second considers the image as a whole, to understand the change in image statistics.

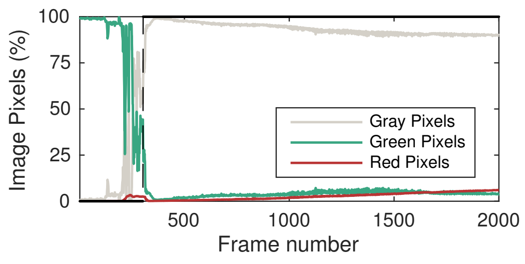
In the following, the results of all the tests are presented side-by-side, since it facilitates their interpretation. Yet, by comparing data of images from the two cameras, the effect of the difference in spatial resolution should be taken into account.

Starting with figure 4.10a and 4.10b, these correspond to the first straw example (Test 1) captured with FLIR SC660 and FLIR Vue Pro, respectively. Observing the variation in both graphs, it is evident the ignition provokes an abrupt change in the gray and green levels. However, considering the ignition occurs around frame 300, it is noticeable that the nonradiometric camera responds with some delay.

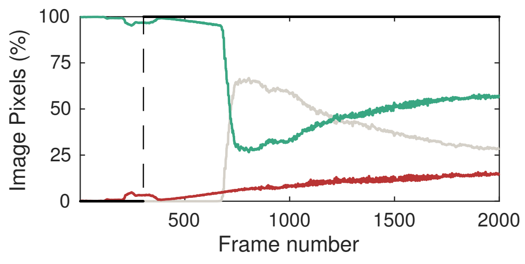
Moreover, examining the graphs from figures 4.10c and 4.10d corresponding to the tent example (Test 2), the change in green and gray levels happens, once again, in an almost symmetric fashion. But, for this instance, the adaptation of the color scale to the fire incident takes effect more slowly. The increase in the percentage of red pixels occurs gradually but still at relatively low levels.

Looking at both of these tests, in which images were captured from a static platform, the variation of the color levels develops in an incremental manner, so it would be difficult to establish a threshold based on the change in consecutive frames. Additionally, the percentage of pixels in gray seems the most promising feature to identify the occurrence of fire. Still, the image does not adjust so abruptly in all cases, so if a low percentage of gray pixels is defined as a threshold, the classification algorithm would be prone to false alarms.

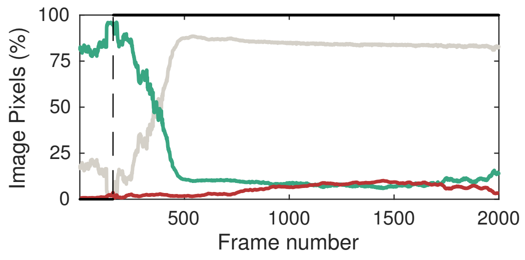
While the results of examples covered so far were obtained from laboratory tests in a controlled environment, under real operating conditions the movement of an aerial platform would cause far greater inconsistency in the color levels. This can be observed by comparing figures 4.10e and 4.10f, which depict the second straw example (Test 3) captured from a static platform and a drone, respectively.



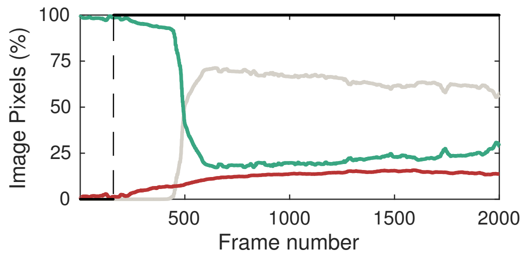
(a) Radiometric: Straw burning (static)



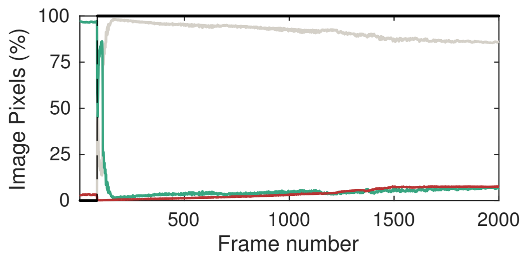
(b) Nonradiometric: Straw burning (static)



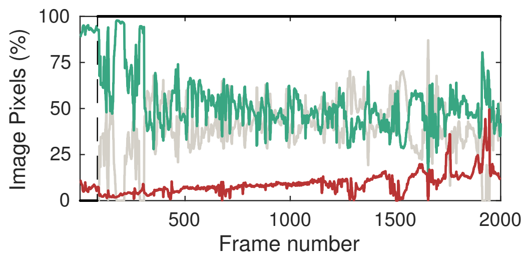
(c) Radiometric: Tent burning (static)



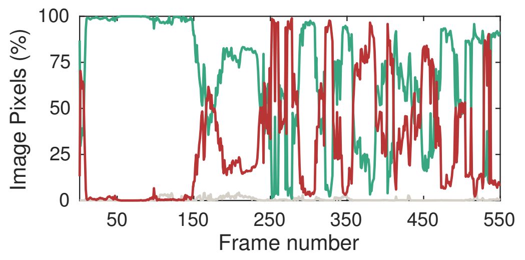
(d) Nonradiometric: Tent burning (static)



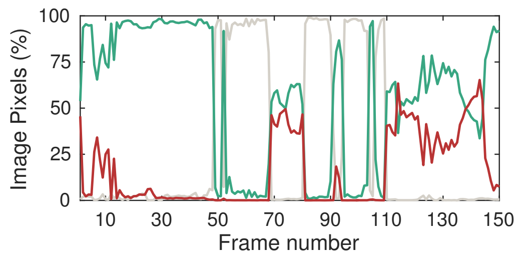
(e) Radiometric: Straw burning (static)



(f) Nonradiometric: Straw burning (drone)



(g) Boom Festival (baseline)



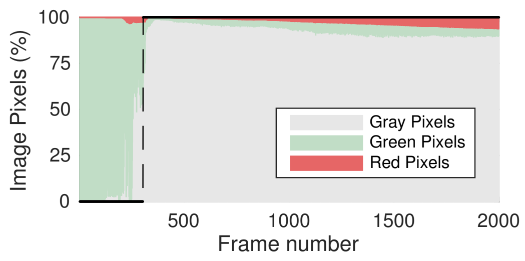
(h) Boom Festival: community kitchen area

Figure 4.10: Color segmentation results.

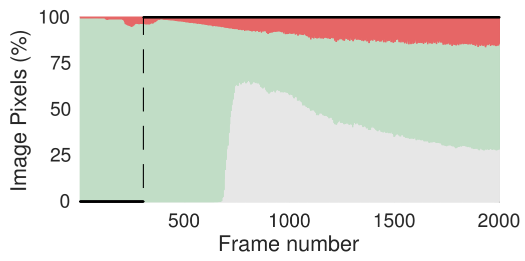
Finally, examining the graphs from figures 4.10g and 4.10h, relative to the tests at the Boom Festival with the tethered balloon, the movement of the camera, as well as the set 5 second time-lapse, have a considerable effect on the sensor response. Now, the variation is no longer incremental, and consecutive frames display abrupt changes in the percentage of either of the segmented colors.

Overall, the principal insight gained from this analysis is that the gray and green percentages adjust in a complementary way, as the color scale adjusts to the hottest objects in the scene. Furthermore, since for fire identification the ignition starts with low spatial resolutions, the red percentage is usually the one with least influence for a fire detection heuristic.

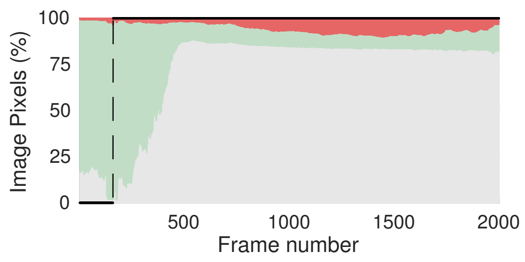
A second approach allows for better intuitions regarding the image statistics. With a different type of visualization, the emphasis is no longer on how the color levels change relative to each other, but rather to the overall aspect of the images. Considering a color segmentation heuristic has to encompass in its thresholds a wide range of scenarios, this enables the overview of percentage ranges for each situation, as can be observed in figure 4.11.



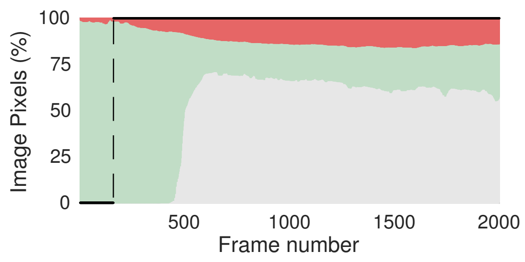
(a) Radiometric: Straw burning (static)



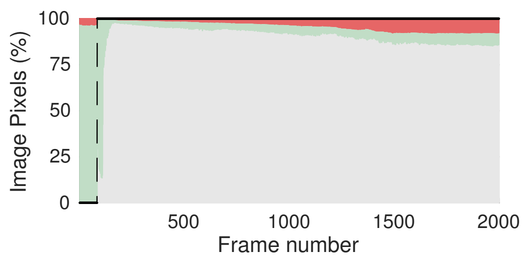
(b) Nonradiometric: Straw burning (static)



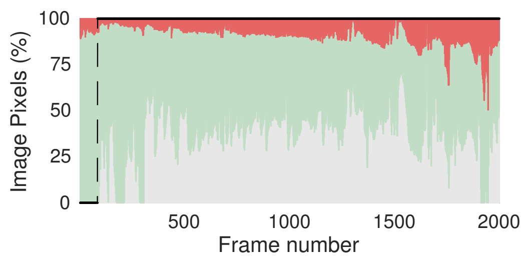
(c) Radiometric: Tent burning (static)



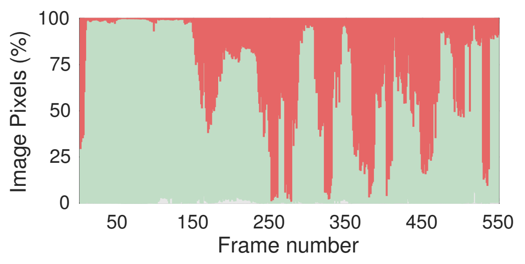
(d) Nonradiometric: Tent burning (static)



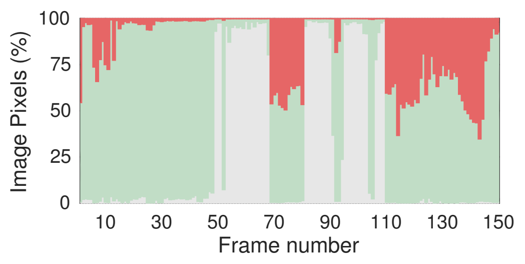
(e) Radiometric: Straw burning (static)



(f) Nonradiometric: Straw burning (drone)



(g) Boom Festival (baseline)



(h) Boom Festival: community kitchen area

Figure 4.11: Color segmentation results.

Observing the results from the static tests, the threshold for gray pixels would have to be established as low as 50%. Otherwise, the case from figure 4.11b would be misclassified. Having in mind that the burning of the tent takes a longer period of time, the gray area should not be the only variable considered. In this case, the gray area reduces due to the increase of red area, and not the green, contrary to figure 4.11e.

Furthermore, the tests from moving platforms, namely the drone and the balloon, demonstrate that a color segmentation heuristic, in this type of application, would have low reliability, if only based on the analysis of thermal imaging data. Figure 4.11e, from a fire ignition in direct view of the camera, and captured at close distance, illustrates the stochastic nature of the variation in the statistics for aerial image capture. Moreover, the reduced values of gray percentage in this case, do not allow for the development of a robust classification algorithm.

Regarding the tests at Boom Festival, which employed a balloon, it was possible to identify that image sequences without fire have negligible percentages of gray pixels, figure 4.11g. But, in opposition, the situations that evidence high levels of gray, figure 4.11h, are not exclusive to a fire incident. Considering the low saturation level of the sensor, further tests would have to be conducted, in order to ascertain the sensor response in such conditions.

4.4 Discussion

The experimental tests presented in this chapter provided several insights into how the thermal camera behaves, in terms of RGB encoding. With the color segmentation devised, the array of situations tested was characterized according to three distinct features, namely the percentage of the three segmented classes: gray, green, and red.

As we have seen throughout this chapter, the variability inherent to dealing with thermal images, which depict relative temperature differences, is a considerable challenge to the development of a robust classification algorithm.

Although a set of heuristic rules could be fine-tuned to work for the situations presented, that type of algorithm would fail, if applied to real-world scenarios. As the tests from Boom Festival illustrated, real conditions account for much higher variability in image content.

Granted that the data from the limited amount of tests is insufficient to build a reliable algorithm at this stage, the data available can still be explored using other techniques. Taking into account the complexity of this classification task, and its feature-rich characteristics, more powerful techniques should be employed in order to achieve better generalization.

To address this issue, the next chapter proposes an intelligent systems approach using deep neural networks. This type of architecture has recently achieved state-of-the-art results in computer vision tasks. These models are prepared to handle high-dimensional data, and spatial features of high complexity. Hence, the approach will depart from thermal imaging, and deal primarily with color images without prior feature extraction.

Chapter 5

Deep Neural Networks

While for a long time artificial intelligence (AI) was a topic restricted to scientific circles, lately the subject has received much attention from mainstream media. With the hype thriving on buzz words like self-driving cars, smart personal assistants, and delivery drones. Despite that some overhype is brought by the nature of the social media world we live in, the attention to this field is very much warranted. In recent years, deep learning, a subset of AI, has been breaking records in scientific disciplines like computer vision [30], and natural language processing [31]. On the one hand, AI is paving the way for instrumental breakthroughs in the areas of autonomous vehicles and robotics, which will soon propel disruptive advances in the automotive industry [32, 33]. Also, new industries are emerging altogether with service drones estimated to become a near 20-billion-dollar market by 2022. The increased complexity of tasks requires models that can handle uncertainty and adapt to environment changes (i.e. obstacles, disturbances). In that regard, deep neural network models, have achieved outstanding generalization in classification tasks, and there is a growing interest in leveraging this type of learning ability to object detection problems [34, 35, 36].

Deep neural networks (DNNs), in a modeling sense, are a class of nonlinear models capable of learning multiple levels of abstract representations from raw data without the use of expert knowledge. These models can have different learning schemes, but regardless all make use of a set of methods known as deep learning, or, on a broader scope, machine learning.

Deep learning is an emergent branch in the field of AI, that has gained a lot of prominence in the last few years. Its methods are based in representation learning and rely on stacking of simple nonlinear modules to uncover multiple levels of representation. Starting from the raw input, features are mapped to increased levels of abstract representation, to ultimately generate an output for classification [37].

Unlike classic neural networks, which require prior feature extraction, DNNs handle the feature extraction within the model. The latter obtain the *deep* denomination as a result of stacking multilayers, and are capable of dealing with raw high-dimensional data, i.e images, text, sound, or video. By comparison with models derived from first principles, this type of model is known as a black-box.

In what concerns interpretability and transparency, black-box models are considered opaque, and once data enters the system, there is a loss of meaning. Notwithstanding, while this could be considered an obstacle to the broad use of NN in many applications, it is not. As noted by Langley and Simon, problems ranging from design, control or planning can be divided into a sequence of simpler classification or prediction tasks, thus opening up the span of applications that can be tackled [38]. To illustrate this reality, figure 5.1 depicts the interconnections of the broad applications of AI with its respective domains and methods.

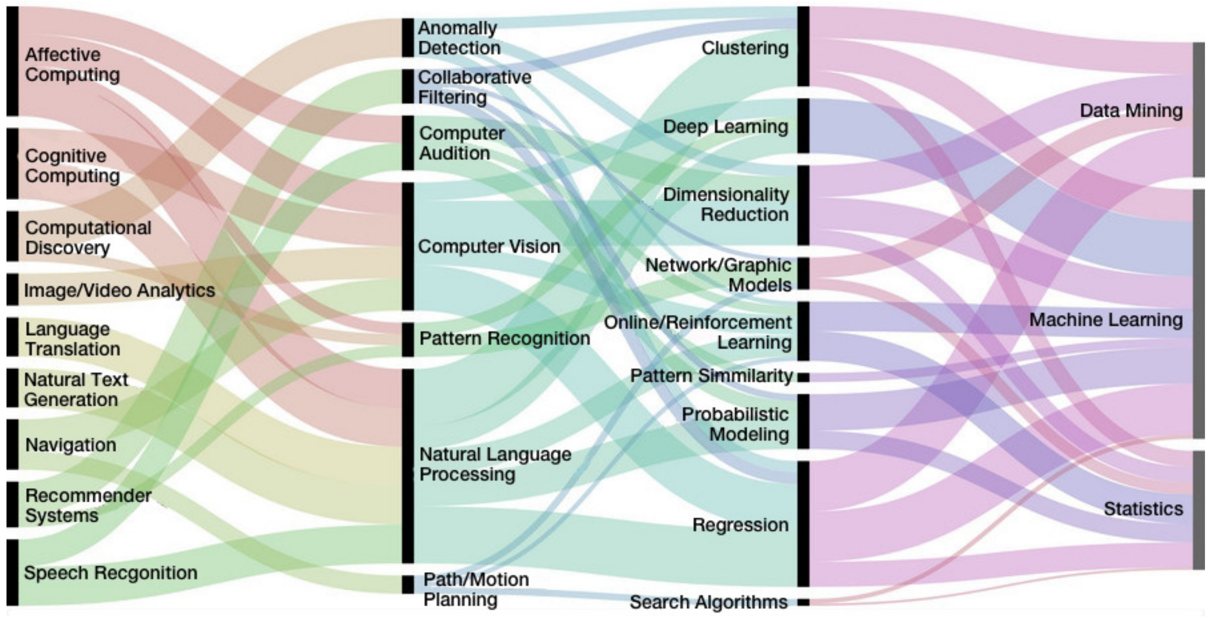


Figure 5.1: Artificial Intelligence: applications, domains, methods[1]

As becomes evident from the preceding graph, disciplines in this field are highly intertwined, and several applications are subject to a varied array of different approaches. In particular, neural networks, a cornerstone of artificial intelligence, has a substantial role in the machine learning stake represented in the graph above. Front and center, also comes the relevance of Computer Vision (CV) in specific related to image and video analytics.

Albeit the fact that black-box models do not give insight into why, in turn they excel at answering the questions of what and when, and that output in and of itself can be valuable in many different contexts. In the case in question, the point of using a black-box model is precisely to leverage these capabilities in a task that requires classification of high-dimensional data, and for which it would be unfeasible to establish efficient algorithms for large amounts of data.

After this introduction, this chapter is organized as follows. First, section 5.1 begins by presenting the state-of-the-art of computer vision using artificial neural networks focusing on the recent architectural advances. This is complemented with an overview of the current landscape of deep learning frameworks followed by a more centered framing of the scientific contributions regarding the problem of video classification. Then, section 5.2 focuses on convolution neural networks with the explanation of the building blocks of this type of architecture. Subsequently, section 5.3 introduces the Inception V3 model in detail, and ultimately in section 5.4 the optimization and backpropagation algorithms are presented.

[1] Adapted from: http://blog.luxresearchinc.com/blog/2015/10/2848/

5.1 Neural Networks for Visual Recognition

A neural network (NN) is a nonlinear model built to mimic the way the human brain processes information, and is composed by a network of parallel and distributed computational elements called neurons. These data-driven models gather their knowledge, depending on their learning scheme, either through detection of patterns by mapping inputs to outputs (i.e. supervised learning, reinforcement learning), or by finding relationships in the data provided (i.e. unsupervised learning).

Historically, the basis for these models originated with the model of a neuron proposed by McCulloch and Pitts [39], that later evolved to the *perceptron* model introduced by Rosenblatt in 1957 [40]. Subsequently, emerged the multi-layer perceptron (MLP), a feedforward network that unlike the perceptron is capable of achieving good generalization for nonlinearly separable patterns.

From early on, being visual perception such an integral part of how we sense the environment around us, translating NN understanding of brain function urged considerable interest from the computer vision field [41]. In 1980, Fukushima proposed an unsupervised multilayered network known as the *neocognitron*. It demonstrated that a hierarchical organization of the layers of the network gave the ability to learn features invariant to distortion and position shifts, and introduced the idea of sharing weights in patches to obtain translation invariance across the image [42]. Inspired by this concept, LeCun proposed *LeNet-5* and established the term convolution neural network. *LeNet-5* comprises convolutional and pooling modules for feature extraction, fully-connected layers, and a softmax classifier. LeCun et al. demonstrate this architecture can be successfully trained using the backpropagation algorithm for a hand-written digit recognition task [43].

Although convolutional architectures showed promise in the domain of computer vision, the increased difficulty in training networks of higher complexity, coupled with the failure to deliver practical results, motivated a slower development in this investigation area. For an extended period, in the field of computational intelligence, the scientific community favoured approaches such as SVM, and fuzzy rule-based systems in detriment of neural networks.

The resurgence of the interest in neural networks came about in 2006, as a result of continuing work by a group of researchers brought together by the Canadian Institute for Advanced Research (CIFAR), such as Hinton, Bengio and LeCun. Amongst other successes, the implementation of layer pre-training under an unsupervised learning procedure introduced an effective way for training deep neural networks.

More recently, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been the stage for many breakthroughs in the field of computer vision [44]. The annual competition started in 2010 and is a benchmark for state-of-the-art computer vision algorithms, in image classification, single-object localization and object detection. The ImageNet dataset consists of over 15 million labeled high-resolution images in over 22000 categories [45]. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories, adding up to 1.2 million training images, 50000 for validation, and 150000 for testing.

Ever since AlexNet, a deep convolutional neural network, won the contest by shattering previous records [30], models have expanded in depth. The model proposed a combination of strategies to make training faster using Rectified Linear Units (ReLUs) instead of sigmoidal activation functions, and leveraged GPU parallel computation for time efficient training. Additionally, to reduce overfitting the model employed data augmentation techniques in order to generate more training examples, and a new regularization method called dropout. This marked a pivotal point in this field, so much that most of these techniques would become a standard for training deep learning models.

On the footsteps of this success, in the following years, the top ranks have been dominated by other CNN models. Notably, in ILSVRC 2014, the evolution to deeper and wider networks yielded significant improvements in top-k accuracy[2] with a close competition between the winner GoogLeNet, submitted by a team from Google, and the runner-up VGGNet from Visual Geometry Group of the University of Oxford.

The VGG model, proposed by Simonyan and Zisserman [46], demonstrated that discriminative power improved with a deeper network, and that a repetition of simple building blocks like 3x3 convolutions and 2x2 pooling layers were enough to achieve a good hierarchical representation of image data.

In turn, GoogleLeNet [47] main innovations centered on solutions for better computational efficiency, only needing 5 million parameters. This represented a twelvefold reduction with respect to its predecessor AlexNet, which used 60 million parameters. In comparison, the performance gains of VGGNet increased computational cost, since it employed about three times more parameters than AlexNet. GoogleLeNet, codenamed Inception, has suffered several extensions since with considerable improvements in top-1 and top-5 accuracy.

The continuing trend is for deeper models, and in 2015 for the first time, a network surpassed the performance of the human reference for this challenge with a top-k error rate of 3.6%, with ResNet proposed by Microsoft Research Asia [48].

To illustrate the recent developments in deep convolutional architectures the comparison between models can be observed below in figure 5.2.

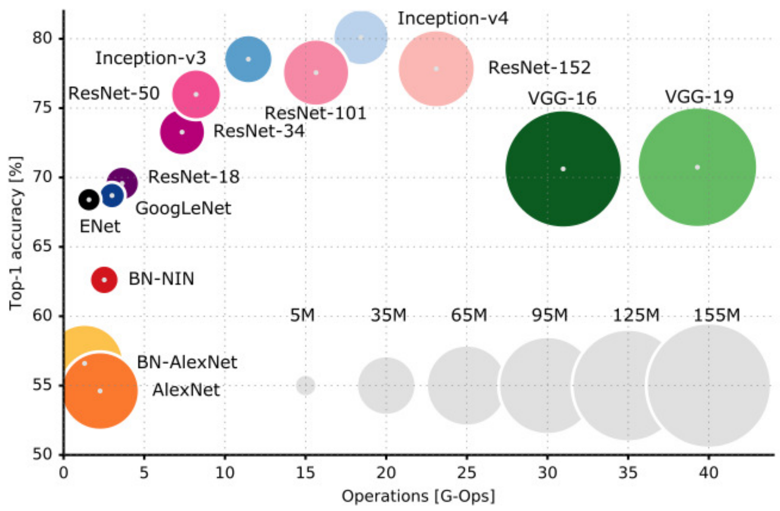


Figure 5.2: Comparison of DNN models [49].

[2]Top-k accuracy, or top-k error, are performance measures widely used to evaluate multinomial classifiers. For instance, top-1 accuracy refers to the rate at which the model correctly predicts the class of an image, while top-5 accuracy relates to a model attributing the correct label within its top 5 predictions.

5.1.1 Deep Learning Frameworks

In recent years the advances in novel neural network architectures have been concomitant to the development of new machine learning programming libraries that have supported and boosted the momentum deep learning has gained.

Machine Learning (ML) algorithms are highly regarded for yielding state-of-the-art results in a wide array of problems, including computer vision, as we have just seen. Nonetheless, this type of algorithms requires a daunting volume of calculations, making them viable only if computing resources are properly and efficiently optimized. For that reason, there was a striking need to evolve the computing methods employed, which motivated investigation in the areas of numerical computation and machine learning. These efforts have received the continuous support of renowned graphics card manufacturer NVIDIA over the years. Figure 5.3, depicts some of the early prominent programming libraries.

Figure 5.3: Deep learning libraries

Torch is a framework for scientific computation that implements a vast range of machine learning algorithms providing an efficient environment for deep learning research in general, and particularly computer vision [50]. Originally released by the Idiap Research Institute in Switzerland back in 2002, today it is widely used and developed at tech companies like Facebook, Google Deepmind, and Twitter.

Theano, is a Python library to target computationally intensive scientific research, and consists of a mathematical compiler that optimizes the evaluation of symbolic expressions for CPUs and GPUs [51]. First introduced in 2007, it was created at the Montreal Institute for Learning Algorithms (MILA) in University of Montreal, led by Yoshua Bengio.

Caffe is a deep learning framework that originated in the Berkeley Artificial Intelligence Reseach Laboratory (BAIR) in late 2013, and was developed specifically for computer vision applications by leveraging the use of CNNs [52]. While the lack of a more general purpose implementation presents as a disadvantage when compared with other DL libraries, the comprehensive set of pre-trained models makes it excellent for working on computer vision problems.

Although up to a point, most ML/DL libraries emerged from research in Academia, lately the interest from technology companies like Google and Facebook triggered the need for frameworks better adjusted for handling big-data requirements efficiently and capable of operating in mobile platforms just as well. Moving forward, there are three novel frameworks that are likely to dominate the AI field, namely, Caffe2, PyTorch, and Tensorflow, figure 5.4.

Figure 5.4: Novel deep learning frameworks.

Considering *Caffe* had initially been created to tackle computer vision applications, it had the limitation of not being suited to address recursive deep learning applications with inputs like text, sound, video or time series data. This caveat has since been overcome with the open-source release of *Caffe2*, by Facebook in April 2017.

Caffe2 extends the previous library not only in respect to the domain of applications that can be targeted (i.e. speech recognition, medical, IoT), but also in what regards training techniques at the disposal of developers. This updated framework also delivers a toolkit with enriched features such as large-scale distributed training prepared to perform on cloud-computing and mobile environments.

PyTorch pivots Torch implementation from Lua programming language to Python which is a huge step since it facilitates integration with other Python libraries. Released by Facebook in January 2017, at the time of this writing it has gained considerable uptake both in tech companies, i.e., Twitter, NVIDIA, and in academia, in universities like Stanford, Carnegie Mellon, Oxford, among others.

On the one hand, the strategy Facebook devised to extend two of the most well established DL frameworks has different objectives in mind. Whereas *PyTorch* is more tailored for research and development purposes, with *Caffe2* the aim is to focus on portability while maintaining scalability and performance, even leaning towards deployment of mobile solutions with systems like *Caffe2go*.

On another hand, Google took a different approach building a unified framework altogether to bridge the gap between research oriented development libraries and platforms ready for the massive data analysis requirements that are the backbone of all Google services.

Tensorflow implements a dataflow graph to represent numerical computations, which makes it more flexible to deploy the computational workload to several workers (CPUs, GPUs, servers) [53]. With that, it naturally integrates well with cloud computing platforms, being specially built for distributed parallel computing solutions.

Until now, the exposition gave a concise overview of the state-of-the-art in what regards the various DL frameworks. In what concerns this work, the framework that will be used is TensorFlow, in particular the 1.3.0 version of its Python API. This framework enables working with pretrained models, namely the Inception V3 model trained on the ImageNet dataset. The approach developed in this work proposes an application of Inception V3 to a fire identification problem by employing a transfer learning method.

5.1.2 Transfer Learning

Transfer Learning (TL) is a machine learning method that consists of leveraging a previously optimized network, a pretrained model, and use it to learn an entirely new task, using a different dataset. Pretrained models are extensively used for tasks very different than the ones for which the network was originally trained. In figure 5.5, a generic scheme is given of this type of technique. For instance, for a CNN model, the feature extraction (FE) part already differentiates which spatial features are relevant, so it is likely to perform well on a new task. The parameters for FE optimized in the original task, are reused in the new task, leaving only the classifier to be retrained to map the new set of inputs to their respective labels (outputs).

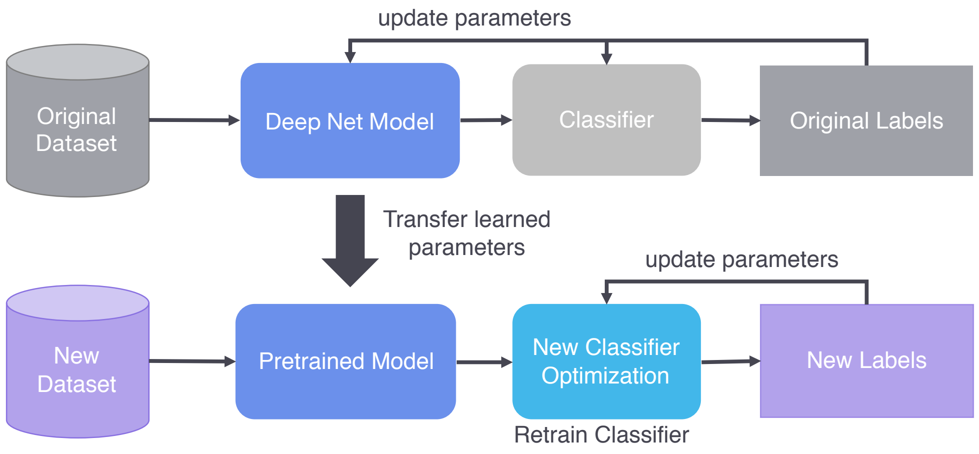


Figure 5.5: Transfer Learning scheme.

As seen so far, much of the top-tier results achieved using DL were accomplished on large datasets, e.g., benchmark datasets such as ImageNet, CIFAR, MNIST. This point is critical to why DL models can work so well, and why these models have become versatile. On the one hand, since neural networks learn from examples they inherently become more accurate and robust when they have a larger set to learn from. On the other hand, having a comprehensive input space is also of concern because models are more likely to fail in the testing stage when they are given new inputs out of the input space. Therefore, when working with supervised learning models, it is imperative to be aware that these models excel at generalization but are not capable of extrapolate for data for which they have not learned to extract representations.

Having highlighted the use of large datasets, it is important to address that the quest for artificial intelligence is completely dependent on data. Although the potential applications of DL algorithms are growing by the day, large datasets often take years to build, and even on a small scale, data acquisition depends on the collaboration of stakeholders from different fields. Moreover, while data is increasingly becoming a 21st century commodity, there are still many challenges regarding data collection, compliance with privacy laws, and open-access. To mitigate such limitations, several DL techniques can be employed including transfer learning and data augmentation.

As deep neural networks grow wider and deeper, the amount of parameters to be optimized skyrockets and training becomes an extremely cumbersome process, computationally and time-wise. As a matter of fact, in many cases, retraining a network to its full extent not only could be a waste of time and computational resources, but it also can lead to inferior results. In that sense, starting with a model that has achieved state-of-the-art results for several different datasets guarantees good generalization, therefore making it capable of extracting good representations when applied to new data.

Besides enabling working on problems with far less data, transfer learning also ensures the resulting model is more robust. In this work this technique will be applied by employing one of the models mentioned previously in figure 5.2, namely the Inception V3 model. Later on, in section 5.3 the model architecture will be described in detail, as well as further characteristics that make it adequate for this problem.

5.2 Convolution Neural Networks

This section focuses on architectural aspects of convolution neural networks, starting by relating concepts from the domain of computational vision to neural network implementations. Then, the building blocks of these networks are presented, with the introduction of five different types of layers: fully-connected, convolutional, two pooling layers (max and average), and the dropout layer.

Despite being conventionally called CNNs, this denomination comes strictly from a semantic sense, which derives from what is typically referred to as a convolution in the computer vision domain. In this field, the term convolution encompasses the notion from the Latin word *convolvere* (*con* – "together" + *volvere* – "roll"), and refers to the process where a filter is combined with a window of the input image, hence resulting in a filtered image.

In a convolutional layer of a neural network, the neurons are the weights of the filter matrix, and the filtered output maps the layer activations. These are denominated as activation maps, but are also commonly referred to as feature maps, since they perform feature extraction. In deep neural networks, convolutional layers do not employ a single filter but rather a stack of them, giving the layer its depth. Figure 5.6 illustrates this concept by presenting a CNN composed by two convolutional layers with stacks of filters. The last two vertical blocks represent a fully-connected layer and a classifier, respectively.

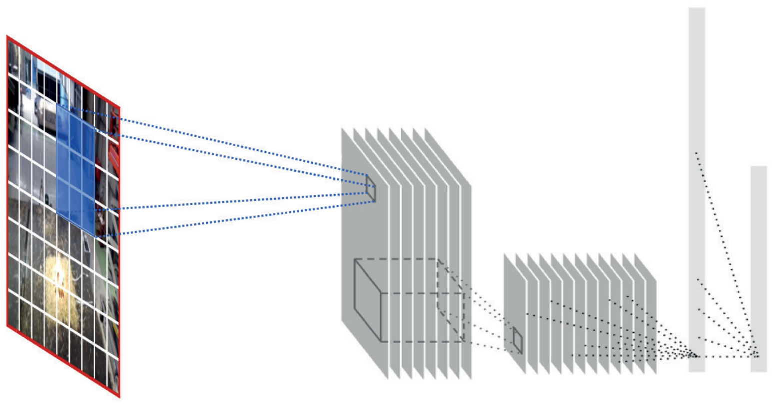


Figure 5.6: Example of a simple convolutional neural network.

Although neural networks are black-box models, in CNNs the weights being optimized correspond to the parameters of the linear filters being applied to the input images. Meaning the values of these weights, which are responsible for the filtering process are not without significance. Thus, since the filtering is a cornerstone of the feature extraction process, it is important to understand its underlying mathematical operations, even if only for a matter of preciseness.

While in CNNs the terms filtering and convolving are used interchangeably it is important to note that the mathematical operation performed is not quite a convolution. In practice, when applying linear filters to the input images the operation performed is not computed as a mathematical convolution but rather a cross-correlation. The next section delves deeper into this issue to establish some theoretical aspects regarding convolution so that this distinction becomes clear.

5.2.1 Convolution

From signal processing theory, for a continuous-time Linear Time Invariant (LTI) system, the output signal, $y(t)$, is defined as the convolution of the input signal, $x(t)$, and the impulse response $h(t)$. Then the convolution integral is given by:

$$y(t) = \int_{-\infty}^{\infty} h(\lambda)x(t-\lambda)d\lambda \tag{5.1}$$

in compact form,

$$y(t) = h(t) * x(t)$$

where $*$ symbolizes the convolution operator.

Discrete Convolution

Because image data has a discrete structure, in this explanation it makes sense to extend convolution to the discrete-time domain, which can be written as follows:

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{5.2}$$

where the output signal is represented by y, the input by x and the impulse response by h.

2D Convolution

In computer vision, to perform a convolution the previous operation has to be extended to the two-dimensional domain. First, the filter matrix is flipped by performing a rotation of 180^{o}, and afterwards the weighted sum of the product of the kernel entries with the input window from the image is computed. Considering an input image, I, and a filter kernel, K, both to be 2D matrices, the convolution is given by:

$$I * K = S(I * K)(i,j) = \sum_{m}\sum_{n} I(i-m, j-n)K(m,n) \tag{5.3}$$

where the indices m and n correspond to the pixels of the image, which is of size (m x n).

However, training a CNN poses a nonconvex optimization problem whose objective is to find weight matrices that map the input to the correct output. While these weights are not meaningless, the computation relative to the reversing of the filtering can be spared, saving millions of calculations throughout the network by eliminating this step. Taking this change into account, then the filtered output will be calculated with a mathematical operation very similar to convolution known as cross-correlation.

2D Cross-correlation

In the 2D space, cross-correlation can simply be achieved by the sum of the point-wise matrix multiplication between the input image window, I and the filter kernel, K.

$$I \star K = S(I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{5.4}$$

where $\star$ denotes the cross-correlation operator.

Taking a step back to the one dimensional space, we know that convolution and cross-correlation operations are equivalent in the particular case of even functions, where there is symmetry with respect to the y-axis. Conversely, in the 2D space, by projecting an even function in both directions, x and y, the result will be a matrix, with the same dimensions that represents the 2D kernel.

Example

To illustrate, we will use an even function, in this example a Gaussian function, $g(x)$, defined as:

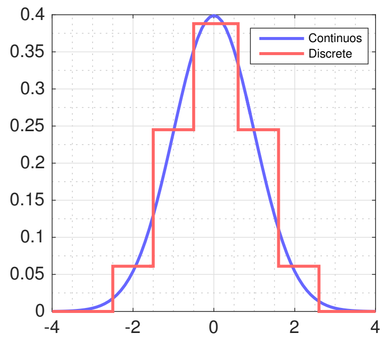
$$g(x) = \frac{1}{\sigma_g \sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu_g}{\sigma_g}\right)^2\right] \tag{5.5}$$

where, σ_g is the standard deviation and μ_g the mean value of the function.

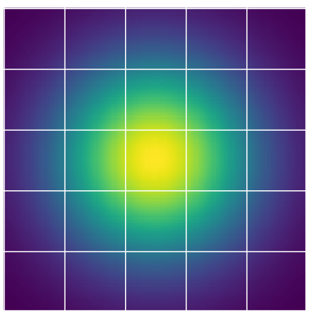
By means of matrix multiplication, the function $g(x)$ is projected in the x and y planes, resulting in the function $g(x,y)$ given by:

$$g(x,y) = \frac{1}{2\pi\sigma_g^2} \exp\left(-\frac{x^2+y^2}{2\sigma_g^2}\right) \tag{5.6}$$

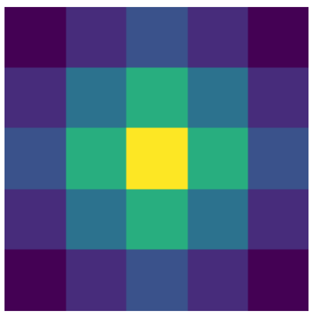
If we for instance assume a mean value of 0 and standard deviation of 1, equation (5.5) yields the curve depicted in 5.7a. For the projection of the function in 2D, now using (5.6) the results are the ones illustrated in figure 5.7b and 5.7c, continuos and discrete, respectively.



(a) 1D Gaussian



(b) 2D Continuous



(c) 2D Discrete

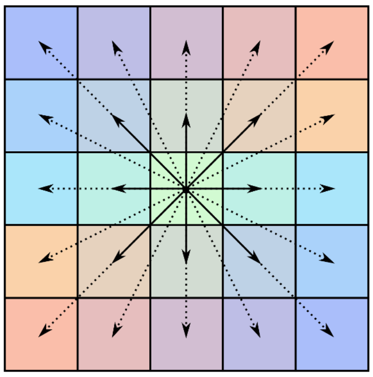
Figure 5.7: Gaussian Function.

Using the 5 x 5 filter built before, the discrete values would be as displayed in figure 5.8a but multiplied by a factor of 1/273 (the decimal value of the kernel inputs are factorized to ease visualization).

Furthermore, observing the matrix entries reveals that a kernel built with even functions results in a centrosymmetric matrix, figure 5.8b.

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

(a) Gaussian kernel



(b) Centrosymmetry

$K_{(0,0)}$	$K_{(0,1)}$	$K_{(0,2)}$	$K_{(0,3)}$	$K_{(0,4)}$
$K_{(1,0)}$	$K_{(1,1)}$	$K_{(1,2)}$	$K_{(1,3)}$	$K_{(1,4)}$
$K_{(2,0)}$	$K_{(2,1)}$	$K_{(2,2)}$	$K_{(2,3)}$	$K_{(2,4)}$
$K_{(3,0)}$	$K_{(3,1)}$	$K_{(3,2)}$	$K_{(3,3)}$	$K_{(3,4)}$
$K_{(4,0)}$	$K_{(4,1)}$	$K_{(4,2)}$	$K_{(4,3)}$	$K_{(4,4)}$

(c) Weight matrix

Figure 5.8: Filter Matrix.

In this example, where the same function was used for the projection in both dimensions, besides being centrosymmetric the kernel is also bisymmetric, but that is not relevant to the point in question. The Gaussian filter was chosen for this example, despite being bisymmetric and not only centrosymmetric because the spatial filters commonly used in CV have an isotropic structure, e.g. Laplacian filters. Toobtain a filter that is only centrosymmetric, it would suffice to use one even function for the projection in x, and a different even function for the projection in y. An example of this would be the Gabor filter, that uses a Gaussian function, and a cosine function.

Moreover, in figure 5.8c, in order to better visualize symmetry with respect to the center of the matrix, weights, $K_{(i,j)}$, with the same values are signaled by being assigned the same color, so that $K_{(0,0)} = K_{(4,4)}$, $K_{(0,1)} = K_{(4,3)}$, and so forth.

In these conditions, the convolution would result in:

$$I_{(0,0)}K_{(4,4)} + I_{(0,1)}K_{(4,3)} + ... + I_{(4,4)}K_{(0,0)} = I_{(0,0)}K_{(0,0)} + I_{(0,1)}K_{(0,1)} + ... + I_{(4,4)}K_{(4,4)}$$

$$\therefore \quad I * K = I \star K$$

This simple example demonstrates that only when the kernels have a centrosymmetric structure, the convolution and cross-correlation are equivalent.

Nevertheless, as alluded before, in CNNs since the filter parameters will be computed iteratively over the optimization process, the kernels will not have a centrosymmetric structure. Therefore, in a purely mathematical sense, when applying a linear filter the operation is in fact cross-correlation, because neither the conditions for equivalency between convolution and cross-correlation are verified, nor the filters are being flipped before point-wise matrix multiplication.

This can seem like a detail, but having discussed the large amount of parameters necessary to implement deep neural models, computation wise this aspect plays a crucial role.

5.2.2 Fully-Connected Layer

The structure of the fully-connected layer, is based on the architecture of classic neural network models like the multi-layer perceptron. In fully-connected layers, the neurons in adjacent layers have pairwise connections, but despite their name, neurons within a single layer do not share connections between them. To establish the mathematical formulation used throughout this work, the next example illustrates the connections between neurons for this type of connection scheme.

Consider a network or layer with d inputs, n hidden neurons, and 2 outputs, as illustrated in figure 5.9.

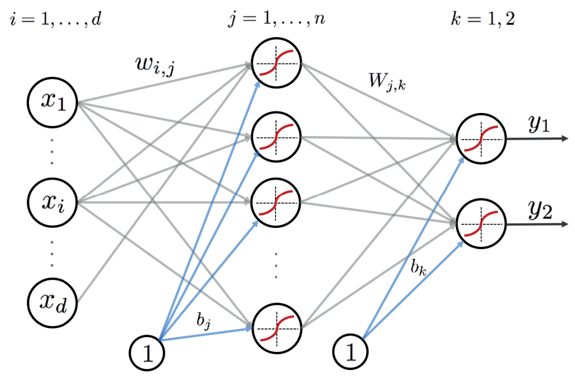


Figure 5.9: General neural network model with an input layer, a single hidden-layer, and an output layer.

In figure 5.9, the grey arrows represent the weight of each connection, and the blue the biases. Between the input and the hidden layer the weights are represented by $w_{i,j}$ while for the connections between the hidden layer and the output layer, these are denoted as $W_{i,j}$. This difference in notation is used since it facilitates distinguishing the two when we discuss error-backpropagation in section 5.4.3.

For each neuron, the input, z, is computed in matrix form as the dot product between the weights of the connections, $\boldsymbol{w}$, and the inputs, $\mathbf{x}$, plus a bias term, b:

$$z_j(\mathbf{x}) = \boldsymbol{w}^\intercal \cdot \mathbf{x} + b = \sum_i w_{ij} x_i + b_j \tag{5.7}$$

The output of a neuron, h, is calculated using the input z_j and an activation function denoted as f:

$$h(\mathbf{x}) = f(z(\mathbf{x})) \tag{5.8}$$

Then, the input to the output layer will be expressed as:

$$z_k(\mathbf{x}) = \boldsymbol{W}^\intercal \cdot h(\mathbf{x}) + b \tag{5.9}$$

Finally, using σ to denote a sigmoid activation function for the output layer, the predicted output is given by:

$$y(\mathbf{x}) = \sigma(z_k) = \sigma(\boldsymbol{W}^\intercal \cdot h(\mathbf{x}) + b) \tag{5.10}$$

Having established a generic notation for the computations in neural networks, it is important to note that due to their amount of connections FC layers are very computationally expensive. Therefore, these are used sparingly in CNNs, and usually at the penultimate layer before the classifier.

5.2.3 Convolution Layer

Convolutional layers are the main building block of CNNs and are responsible for extracting representations from the data as it flows through the network. They are characterized according to three distinct parameters: filter size, filter stride and the depth. Additionally, for mathematical coherence in matrix multiplication sometimes it is necessary to apply padding. This consists in adding zeros around the borders of the input image.

This type of layer computes the dot product between patches of the input volume and the weights of the filters, as was described in section 5.2.1. Figure 5.10 depicts an input volume with a depth of 3 channels being filtered by a convolutional layer of depth 6 and kernel size of 2x2. The diagram is merely illustrative and padding and stride are not considered. The dimension of the input and output volumes reflects the usual application of this type of layer that reduces data in spatial size and increases its depth.

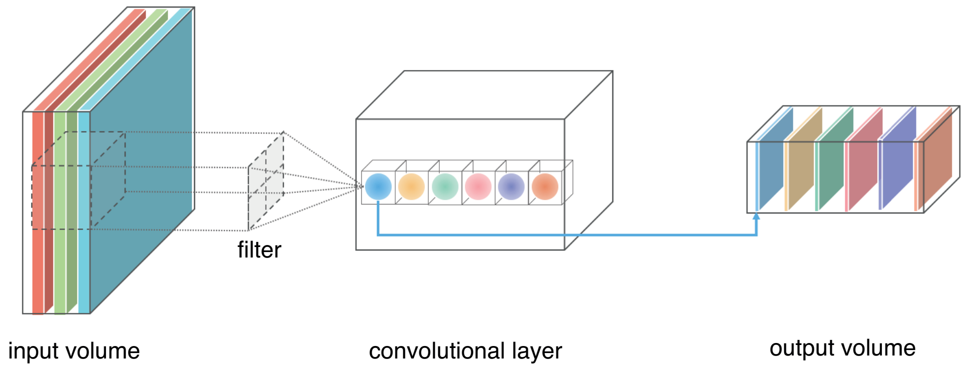


Figure 5.10: Example of a convolutional layer of depth 6.

In figure 5.10 the entire depth of the convolutional layer is considered for a single patch of the input, mapping the input to the output volume of this region. Along the depth, each of the neurons has a independent weight matrix with 12 weights as given by the 2x2 filter and 3 channel depth. The weight matrix that defines the filter of each of the 6 neurons is different, and is used to extract distinct features.

However, neurons for other patches of the image share the same weights for each depth slice, this is known as parameter sharing. To illustrate this, figure 5.11 presents the parameter sharing scheme.

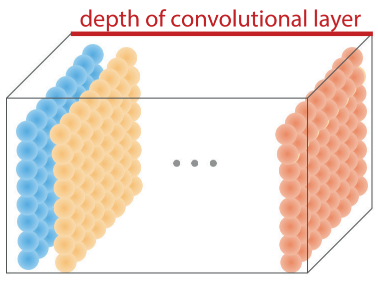


Figure 5.11: Parameter sharing for the slices of neurons in the convolutional layer.

This characteristic of convolutional layer spares millions of parameters and gives the network translation invariance. Since each depth slice in the convolutional layer applies the same filter to every patch of the image, the output of that slice will map a certain type of features. This way, the output volume, in figure 5.10, stacks the activation maps of the different slices of neurons, which extract distinct features.

5.2.4 Pooling Layer

Pooling layers are introduced in the network to perform dimensionality reduction on the activation maps. The previous can scale fast as the stacking of filters increases throughout the network, adding up to a great amount of computations and parameters to optimize during training. The objective of the pooling layer is then to reduce the inputs to smaller activation maps where only the most relevant features are extracted. Typically two types of pooling can be employed, max pooling, and average pooling.

Max Pooling

The output of a pooling layer, u, is defined for the max operation by:

$$u_{ijk} = \max_{(p,q) \in H_{i,j}} z_{pqk} \tag{5.11}$$

where z is the input of the pooling layer, and $H_{i,j}$ area covered by the $K \times K$ kernel. The index k is the depth index of the input volume. Figure 5.12 illustrates the max pooling operation for a single depth layer ($k = 1$) of a 2×2 filter, and stride of 2, which means the filter skips a pixel when sliding over the image.

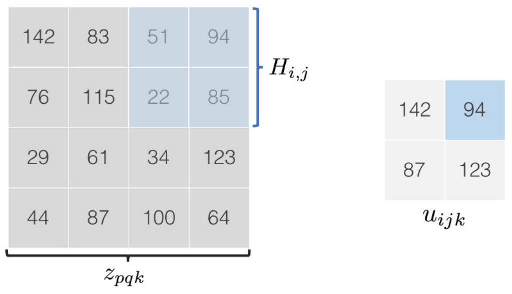


Figure 5.12: Example featuring a max pooling operation with a 2x2 filter and a pooling stride of 2.

Since this approach only retains the strongest activation of each patch of the image, that may present a problem for images with features close together. For instance, if the concern is to identify elements that concentrate spatially in a particular region of the image, that would of course be problematic. Conversely, this type of pooling is also susceptible to respond to outliers, which can be introduced by noise in the image. To avoid a premature loss of information, average pooling is extensively used in alternative.

Average Pooling

Average pooling uses the same variables as max pooling but by averaging each patch of the input, a more general representation is extracted. Now, the output is given by:

$$u_{ijk} = \frac{1}{H_{i,j}} \sum z_{pqk} \tag{5.12}$$

This definition shares the same advantages of max pooling, but is more robust in the presence of outliers.

Pooling puts an emphasis on the relative location of features rather than their exact location, thus reducing the spatial size of the representations which consequently decreases the computational cost.

5.2.5 Dropout Layer

Deep neural networks, due to their complexity, are prone to overfitting when the amount of data available for training is limited. To prevent this from happening, ideally, one would average the performance and test many models to find the optimal parameters for a given task. Though, with limited data and computationally expensive models this is not a feasible option. To address this issue, a recent technique called dropout has proved to be a very effective regularization technique, yielding improvements in model generalization in DNNs.

In essence, dropout consists in dropping a percentage of the hidden units at each training iteration, by zeroing its activations and discarding the influence of the respective connections to the output, as illustrated in figure 5.13.

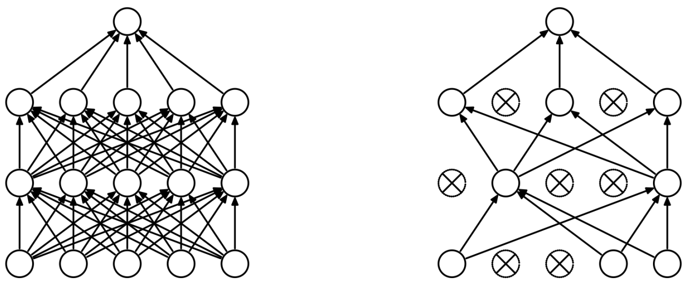


Figure 5.13: Dropout example originally from [54].

This process samples a subpart of the network according to a hyperparameter that defines the probability of keeping the units active. The subnetwork is used to evaluate the training set and update the weights accordingly. After this, for evaluating the validation set, the dropped units are restored before the next iteration. This way a hidden unit cannot rely on the presence of the other hidden units, thus avoiding the co-adaptation to the features of training data. As a result, the model learns redundant ways to map the inputs to outputs, which has the regularization effect of preventing overfitting.

In practice, while tuning the network parameters, part of the learned weights will be discarded at random. At first, it can seem counterproductive since we are sabotaging the learning process on purpose.

The original motivations for this idea have its roots on the sexual reproduction process and conspiracy theories [54]. However, it may be easier to understand this concept by making an analogy to white-box models. By using an example, the causality relationship between this method and its effects also becomes clearer.

For instance, taking the parametric model of a quadrotor derived from first principles, if we do not design controllers capable of handling perturbations such as the wind, in real conditions the system will become unstable, by failing to adapt to changes in the working environment. Following that reasoning, the introduction of a dropout layer in the network is akin to adding a random disturbance to the model. By training the model with this handicap, it inherently becomes more robust. Thus, when the model is presented with new unseen data, it performs better on testing data, due to its better generalization ability.

5.3 Inception v3 Model

In the selection of a deep learning model, its architecture plays an instrumental role. In that sense, having presented the main building blocks of CNNs, this chapter presents the Inception V3 model.

5.3.1 Network Architecture

The Inception v3 model is a deep neural network that like its preceding architectures is built modularly, by means of a stacking of parallel convolution structures known as Inception modules. These employ several different convolution factorization schema depending on their relative depth location in the network and provide different hierarchical levels of feature extraction. These aspects will be explored in detail in the following sections, but for now the focus is on the overall architecture of the network. The full structure of the network is illustrated in figure 5.14, and is composed by several building blocks color-coded as depicted in the legend.

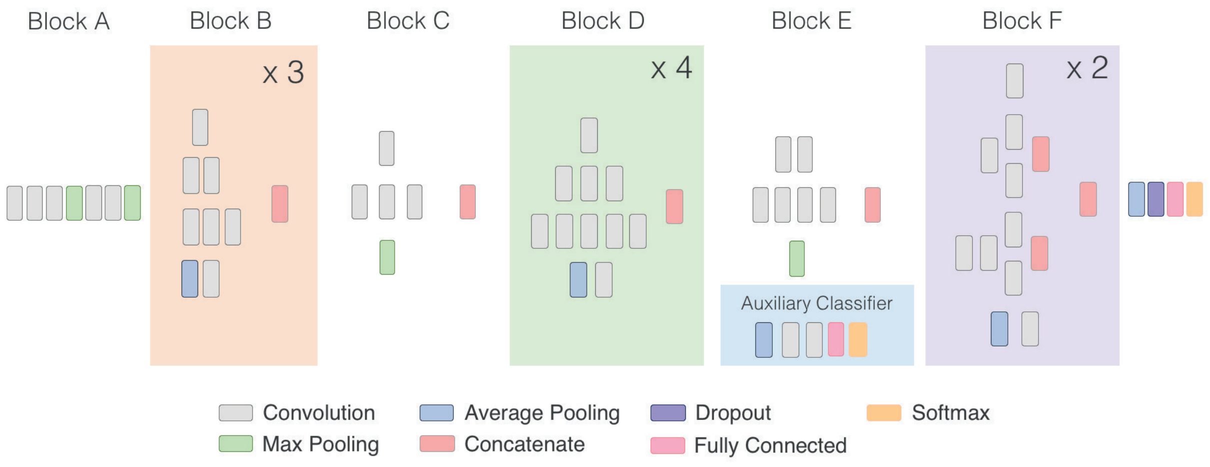


Figure 5.14: Inception V3 network architecture[3]

The diagram with the compressed view of the network divides it in two parts, where the first is responsible for extracting spatial features and the second is a classifier. Most of the extent of the network is dedicated to feature extraction with sequences of different Inception modules to infer features of increasing complexity. In the repeated modules of type D and F there are slight variations in depth, to increase the number of activation/feature maps at higher levels of representation.

The principal advantage of this model is its ability to keep the number of parameters under control due to factorized convolutions and aggressive regularization. When compared to AlexNet, and VGGNet, which employ 65M and 155M respectively, Inception v3 employs only about 35M parameters. This lower computation cost enables the use of this network, in big-data scenarios, as well as in mobile vision scenarios where memory and computational capacity is a constraint.

[3] The diagram presented in the figure is a compressed view of the full architecture. This diagram makes a correction from the representation available in [55]. The error concerns block B, which [56] specifies as a traditional Inception module (without convolution factorization). The referencing of figures in table of [56] does not match up correctly with the information in the text, which is probably the reason for the misrepresentation in the official library documentation [55]. In the latter the code of the model also confirms the traditional configuration, that is correctly depicted above.

5.3.2 Feature Extraction

This section centers on the feature extraction part of the network and the main characteristics of the Inception modules. Inception modules were first proposed in the GoogleLeNet architecture submitted to the ImageNet contest in 2014 [47]. These structures employ convolution factorization to reduce the computational cost of the feature extraction process. In the Inception V3 model, the first three Inception modules (Block B) are a slight variation of the original configuration [47, 56], and can be observed in detail in figure 5.15. The scheme reads from bottom to the top.

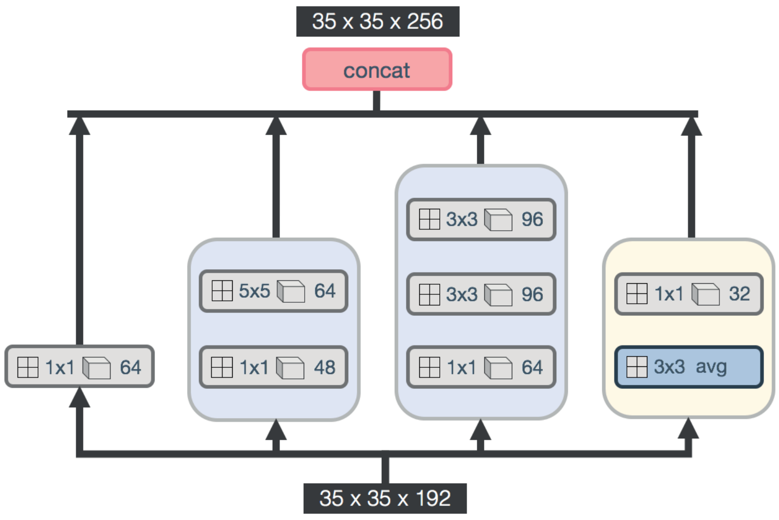


Figure 5.15: Example of the first Inception module of Block B.

Considering the first module has an input of 35x35 and a depth of 192, the data will flow according to the arrows depicted in figure 5.15. In a parallel scheme, the data is convolved by filters of distinct kernel size to extract features as different scales. Since large kernels require a large number of parameters, 1x1 convolutions are used to perform dimensionality reduction of the input. For instance, for the Inception module illustrated in figure 5.15, before applying 5x5 and 3x3 convolutions, the depth of the input to reduced 48 and 64, respectively.

Along the network the dimension of the convolutional layers increases in depth as well as in filter size. Whereas at the initial depth stages these modules extract low level features, towards the end of the network the kernel size of the convolution increases to 5x5 and 7x7, to extract features of higher level. This enables capturing combinations of figures in a single convolution. However, since large kernel sizes would be very computationally expensive the kernels are factorized in $1 \times n$ followed by $n \times 1$ convolutions, as described in [56]. Taking into account a single depth slice of a 5x5 convolution has 25 neurons, the convolution factorization reduces the number of neurons to 10. Hence, having large stacks of filters in each layer, this solution has an enormous effect in the overall computational cost of the network.

This section presented in detail the first Inception module, though since the rest of the modules vary in convolution size as well as in depth, the complete information can be consulted in [56, 55].

While this state-of-the-art network main strength lies in the feature extraction capability, the large amount of neurons necessary for that purpose can make this type of models prone to overfitting.

5.3.3 Hidden Units Activation

Supervised networks use a learning algorithm called error backpropagation, which updates the weights in the direction of the gradient descent of the cumulative error. For that reason, it was believed the nonlinear activation functions had to be differentiable. These functions are inspired in the biological behavior of a neuron when it receives a stimulus.

For an extended period, the use of logistic functions for the activation of hidden units was the most consensual practice. These s-shaped functions, like the sigmoid or the hyperbolic tangent, squash inputs to values between [0, 1] and [-1, 1], respectively. By being continuous and differentiable in their entire domain, these guarantee a bounded output and help prevent exploding gradients during the learning process. However, this also brings some drawbacks from the optimization point of view. For instance, the sigmoid function represented in figure 5.16 reaches saturation at 10 and -10, point from which the weights can no longer be updated.

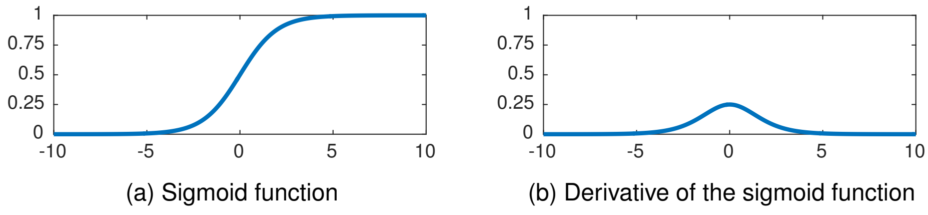


Figure 5.16: Sigmoid activation function.

Additionally, as can be observed in figure 5.16b, the maximum derivative of the sigmoid is 0.25, which means the output error propagated backward would decrease by a factor of 75% on each of the preceding layers. In practice when using bounded logistic functions for activation, shifts in the gradient become increasingly smaller and learning stagnates as a result, this is known as the vanishing gradient problem.

To overcome this issue, rectified linear units are currently the standard activations in hidden layers of recent deep neural models. Besides taming the vanishing gradient problem, this type of activation also allows for much faster training in supervised networks.

Rectified Linear Units (ReLU)

The rectified linear units (ReLU) are elements of the network that provide the nonlinear activation function in the form of a ramp function, as expressed by:

$$f(x) = \max(x, 0) = \begin{cases} x\ , & x \geq 0 \\ 0\ , & x < 0 \end{cases} \tag{5.13}$$

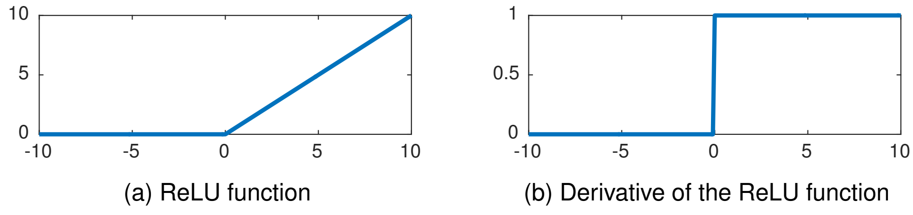


Figure 5.17: ReLU activation function.

While this function speeds up the training process, if a neuron receives a negative input it is maxed out at zero, which means that the neuron dies. Consequently, DNNs usually lose a considerable percentage of their initial neurons during training, but the tradeoff is preferable.

5.3.4 Classifier

The classifier part of the network, identified in figure 5.14, has the function of generating the model predictions. While this part is much simpler than what we have seen thus far, its operations are considerably different. It is composed of four building blocks, in the following order: Average Pooling, Dropout, Fully-Connected and Softmax. The first three were already explained previously in section 5.2 since they are transversal to any CNN model, whereas the softmax is the classifier block that can change depending on the formulation of the classification problem.

In short, after the final pooling layer performs a dimensionality reduction, the dropout layer employs a regularization technique that prevents overfitting. By randomly dropping a percentage of the activations at this stage the network is forced to learn redundant representations. Due to this, the model becomes more robust and is able to improve generalization. This layer is placed immediately before the fully-connected block that generates the outputs before classification. These outputs are often referred to as *logits*, because they represent the unnormalized logistic probabilities that will be used by the last activation function of the model. The latter is known as the score function and maps the *logits* to the predicted class label. To that end, Inception V3 uses the softmax function.

Softmax

The softmax is the score function responsible for generating the predictions of the model. It evaluates the degree of certainty in classification according to a posterior probability based on the *logits* values. As to differentiate from other activation functions referenced previously by $f(z)$, this function is denoted symbolically with $\sigma(z)$ and is given by:

$$\hat{y}_k = \sigma(z)_k = \frac{\exp(z_k)}{\sum_i^K \exp(z_i)} \quad , \quad k = 1, ..., K \tag{5.14}$$

with k representing the index of the class label, for every input, z_k, of the output layer, the softmax will generate K class prediction scores, $\hat{y}$. The numerator acts the same way as a sigmoid function, by squashing the inputs to values between 0 and 1, while the denominator normalizes the output by limiting the sum of all the class probabilities to be bounded between 0 and 1 as well.

For simple fire identification it could be argued that a multinomial classifier is excessive, because the problem can be formulated as simple binary classification. However, recalling the long-term objective of this task is to identify risk situations in camping parks, having them clustered in accordance to their specificity may bring some advantages. It is possibly easier for a model to distinguish 5 types risk situations if they are classified separately, rather than put everything in the same bag and expect the model to be able to learn a more difficult group of representations.

As will be seen in the following section, the selection of score function is closely tied to the choice in objective function of the optimization algorithm responsible for the learning process.

5.4 Optimization

Deep learning algorithms tune the network parameters iteratively using a variety of gradient-based optimization methods. This type of methods are intended to maximize or minimize an objective function, and for the latter, they are usually called cost function or loss function. However, to implement these techniques in the context of a neural network, some important considerations differ from a straightforward optimization problem.

Pure optimization algorithms usually stop when the solution has converged to a local minimum, meaning the gradient has decreased to a stage where the algorithm can no longer improve the solution. Contrary to those, and even to a range of other machine learning methods like regression and SVM, neural networks are nonlinear models, which means the problem departs from convex optimization. Instead, in a DL context, since there is no guarantee of global convergence for a non-convex problem, the optimization process halts when an early stopping condition is satisfied.

This approach, however, is highly dependent on the complexity of the task and is affected by a set of factors that includes hyperparameter selection, weights initialization, and regularization. In addition, what might be advisable from the optimization standpoint may not yield a model with generalization, and for deep networks, this is an area where understanding is still minute [57].

The next section starts by presenting the basis of the optimization process employed in training the network, and the hyperparameters that have to be tuned. Following that, section 5.4.2 covers the loss function selected, and a brief explanation of the backpropagation algorithm is given in section 5.4.3.

5.4.1 Gradient Descent

The problem of optimizing the weights in neural networks employs the error backpropagation algorithm, which is based on gradient descent, and minimizes the error and updates the weights according to a specified learning rate. However, in a problem with high-dimensional data in large quantities such as image classification, it is unfeasible to compute the error for the whole dataset at each iteration. To overcome this challenge, the network is trained in small batches of the original dataset and the error is computed and updated on each training iteration. This enables faster learning in neural networks, and has been proved to be very effective in networks with parameter sharing such as CNNs. This introduces two important hyperparameters that have great influence in the optimization process:

- *Learning rate*: defines the size of the steps in the update of the weights. For high values the steps are bigger but the learning process often stagnates in a bad solution. On the contrary, for small values the improvement between training iterations is smaller, requiring more iterations to reach a good solution. Hence, there is a trade-off between learning rate and the number of iterations necessary to successfully train the model.
- *Training batch size*: sets the amount of images that are used in each training iteration. The value of this parameter depends on memory constraints, but usually is defined as 100 or 250.

The formulation of the optimization problem for adjusting the parameters of a deep neural network consists in finding the weights and biases, that define the mapping between the inputs and the correct outputs. To that end, the objective function will minimize the error between the prediction of the model and the ground truth labels. In this context, this function is denominated as the loss function.

5.4.2 Loss Function

The loss function quantifies the distance between the predicted outputs, $\hat{y}$, and the ground truth labels, y. Typically, the selection of loss function is made in agreement with the score function chosen to generate the model predictions. As seen in the previous section, for the Inception V3 model the activation function of the output layer is the softmax function. For this model the loss is computed using the cross-entropy function. Having defined previously in equation (5.14) the predicted output of the model as:

$$\hat{y}_k = \frac{\exp(z_k)}{\sum_i^K \exp(z_i)}$$

The cross-entropy loss is calculated as the sum of the ground truth labels, represented by $\mathbf{y}$, times the natural logarithm of the prediction scores, $\hat{\mathbf{y}}$:

$$L(\mathbf{y}, \hat{\mathbf{y}}) \;=\; -\sum_k y_k \ln(\hat{y}_k) \tag{5.15}$$

The main reason for choosing the cross-entropy function as a measure of the prediction error is the fact that when using the softmax classifier, the gradient will be bounded between -1 and 1, which is useful for the learning algorithm. Then, the derivative of the loss with respect to the *logits*, z_k, is expressed as:

$$\frac{\partial L}{\partial z_k} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} = \hat{y} - y \tag{5.16}$$

The full deduction can be found in [58].

In addition to the cross-entropy loss, the total loss of the Inception V3 also accounts for a parcel of regularization loss. This model employs a technique denominated as label-smoothing regularization (LSR), to prevent the highest prediction from becoming much larger that the others. If a model attributes large predictions for one class and very small for the others, it becomes too confident of its predictions, and loses the ability to adapt. In that sense, LSR improves learning because it prevents the model from converging for a bad solution at an early point in the training stage. Further details of this method are left out of the scope of this text, and can be read in [56].

Overall, during training, the loss function gives some feedback regarding how training is evolving, and is also a good indicator concerning the adequacy of the hyperparameter selection. Ultimately, before the evaluation and testing stage, the loss for the training set and the validation set will guide the decision if the model was successfully trained or needs further tuning. However, success in training is never a guarantee that the model is sound in terms of generalization, that has to be evaluated using new data.

In the next section, describing the error backpropagation algorithm, the role the loss function plays in the learning scheme is demonstrated, with the description of the equations that update the weights.

5.4.3 Error Backpropagation

The previous sections established that the objective of the optimization problem is to minimize the error in classification, and described the loss function for that purpose. To that end, the weights and biases of the network will have to be updated, in the direction of the gradient descent of the cumulative error. However, knowing the error in classification, to generate a better solution on the next iteration, the influence that each weight has in the output has to be evaluated. This is the principal problem the backpropagation algorithm solves.

The backpropagation algorithm comprises two parts: the forward pass, and the backward pass. In a nutshell, the forward pass encompasses the steps to calculate the output of the network described in section 5.2.2, whereas the backward pass computes the gradients to propagate the error signal and updates the weights of each neuron.

For ease of notation, in this section the bias vector is assumed to be last column of the weight matrix. This can be achieved without loss of mathematical coherence by adding an additional 1 entry to the input vector, $\mathbf{x}$.

Forward Pass

The forward pass of the algorithm computes the activations of each neuron in the network, yielding the values of the inputs of each neuron, z_j, and the outputs, h_j. The procedure has been fully explained previously in section 5.2.2, and illustrated in figure 5.9. This results in the final output of the network, $\hat{y}$, that will be compared with the ground truth labels, y, to compute the error.

Backward Pass

Starting in the output layer, the error in classification is propagated backward through every layer of the network to update the weights. The error signal propagated to each neuron accounts for the influence the weight has in the output error, and is calculated by the successive application of the chain rule. Figure 5.18 depicts this procedure schematically for a single layer. This diagram reads from right to left, starting from the output towards the input layer, and is explained subsequently.

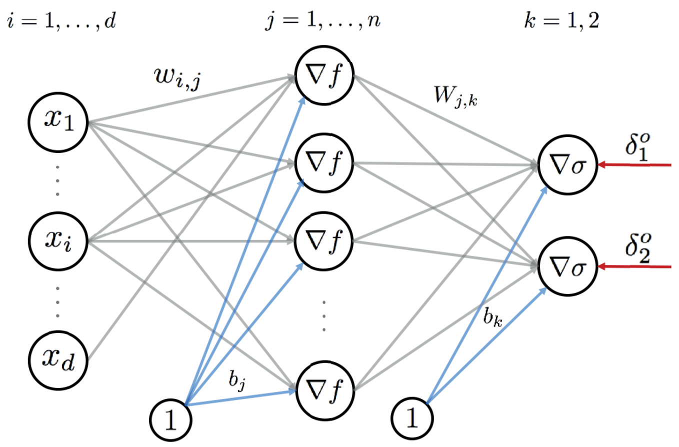


Figure 5.18: General backpropagation procedure for a single hidden-layer network.

The local gradient at the output layer, δ_k^o, represents the error term that is propagated backward throughout the network, and is expressed as:

$$\delta_k^o = e(z_k) \times \sigma'(z_k) \tag{5.17}$$

with σ denoting the activation function of the output layer, its derivative with respect to the input of the output layer, z_n, results in $\sigma(n)(1-\sigma(n))$ for a sigmoid or softmax function. To keep the formula generic, $e(z_n)$ represents the error derived from the loss function, which was introduced in (5.16).

Then, to calculate the update of the weights between the output and the hidden layer, W_{jk}, the loss is derived with respect this variable. Recalling that the input of a neuron in the output layer is given by $z_k = \sum W_{jk} \cdot h_j$, by applying the chain rule:

$$\frac{\partial L}{\partial W_{jk}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} \frac{\partial z_k}{\partial W_{jk}} = \delta_k^o h_j \tag{5.18}$$

The amount the weights are updated takes into account the error and is multiplied by the parameter, η, the learning rate, as:

$$\Delta W_{jk} = \Delta W_{jk} + \eta \delta_k^o h_j \tag{5.19}$$

To propagate the error signal to the hidden layer, and update the respective weights, w_{ij}, the local gradient is computed taking into account the contribution the weights have to the output of the layer, h_j. In the first step, this depends on z_j, the input to the hidden layer. Once again, applying the chain rule:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} \frac{\partial z_k}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} \frac{\partial z_k}{\partial h_j} \frac{\partial h_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}} = \delta_k^o W_{jk} f'(z_j) x_i \tag{5.20}$$

With the error propagated for the neuron in the hidden layer is given by:

$$\delta_j^h = \sum_k W_{jk} \delta_k^o f'(z_j) \tag{5.21}$$

The weights between the hidden and the input layer, will be calculated as:

$$\Delta w_{ij} = \Delta w_{ij} + \eta \delta_j^h x_i \tag{5.22}$$

Finally, to end the backward pass, the weights of all the layers are updated as:

$$W_{jk} = W_{jk} + \Delta W_{jk} \tag{5.23}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{5.24}$$

Having explained the theoretical basis of the learning process, some implementation aspects need to be addressed. This example explains the mathematical operations of a full pass of the backpropagation algorithm. However, the network being considered only has a single hidden layer. For a deep neural network like Inception V3 with a great amount of convolutional layers, the volume of operations would be significantly larger. Considering just the forward pass of the Inception v3 model accounts for approximately 12 x 10^6 operations (fig. 5.2), the computational load is obviously immense. Therefore, applying the chain rule to perform the backward pass would represent a very cumbersome task. The partial derivatives of the backward pass could be calculated explicitly, but that would be a very inefficient process. Instead, TensorFlow uses subexpression elimination to take exact derivatives, and performs automatic differentiation, taking the most advantage of the computations performed whenever possible.

At the beginning of this chapter a great deal of attention was given to the current landscape of deep learning frameworks, in section 5.1.1. Now, having a better sense of the immense amount of computation deep neural networks require, it is clear how essential the contribution these several open-source libraries provide. The optimization of the computations that is performed in the back-end of these platforms is what makes deep learning models viable, both in common personal computers, and in mobile platforms.

With this concludes the presentation of the theoretical aspects of deep neural networks. The following chapter will focus on the implementation of this type of models to a the fire detection problem.

Chapter 6

Implementation

This chapter describes a novel application of the Inception V3 model for a fire identification problem, using TensorFlow. First, in section 6.1, the dataset built for that purpose is described along with the respective partitions that will be employed. Second, in section 6.2, follows the explanation of the preprocessing steps undertaken to capture the frames from the video sources, prepare them to be fed to the DNN. Subsequently, section 6.3 covers the model retraining procedure, hyperparameter selection, and presents the training results. Next, in section 6.4 a selected model is evaluated by testing on a new set of data. At last the results are compared with the results of the heuristic approach presented in chapter 4.

6.1 Firecamp2 Dataset

Having in mind that the acquisition system presented in chapter 2 was designed to capture thermal and color images simultaneously, now the study will center on the latter. Since those experiments account for three video sources only, additional color videos were added to the database to enable training the deep neural network. Then, in addition to the thermal image dataset, an augmented color dataset was created, and some samples can be observed in figure 6.1.

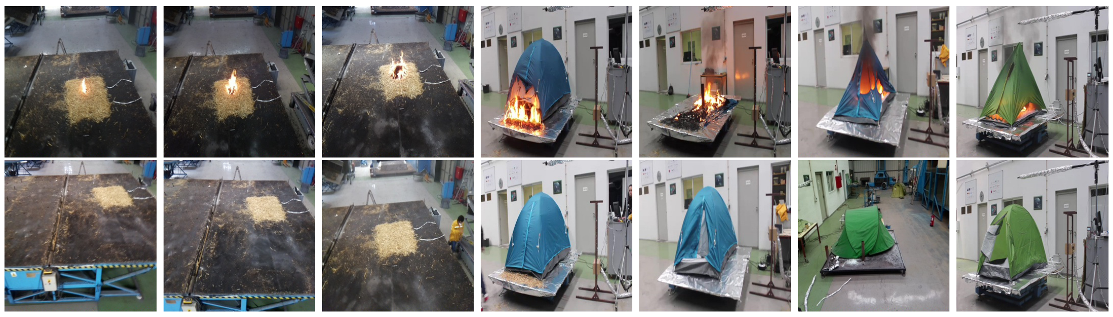


Figure 6.1: Samples from Firecamp2 Dataset.

In the first row of figure 6.1, the images illustrate fire situations, whereas in the second row the images correspond to situations where a fire can not be identified at naked-eye or does not exist at all.

Dataset Partions

The complete dataset is composed of 58266 from 8 distinct video sources, which is a small dataset for a deep learning application. Nevertheless, it is a starting point for evaluating the adequacy of this kind of approach. To identify fire the problem is formulated as a binary classification where one class is *fire* and the second is *not fire*. In table 6.1 the division of the dataset is listed for these two classes.

Table 6.1: Dataset division.

Camera	Fire	Not Fire	Total
Database	30682	27584	58266
Training Data	16000	16000	32000
Testing Data	5143	6407	11550
Straw burning	1699	301	2000
Tent burning	1624	5926	7550
Straw burning (drone)	1820	180	2000

For the training set 32000 frames were selected, from which 20% will be used for validation of the training process. At every 10 iterations the network will evaluate this separate set. For testing, in order to compare the results from this approach with the ones from the segmentation heuristic, now the color images corresponding to the trials studied in chapter 4 will be kept separate on a holdout set. To recap, the first two examples are of straw fire ignition and tent burning when captured from a static elevated platform. In turn, the third example refers to the same test as the first but is repeated to compare the performance for mobile platforms. As done previously, the objective of the testing examples is to detect the point of fire ignition. From these cases a total to 11550 frames were selected for the testing data used to evaluate the classification performance of the network.

6.2 Data Preprocessing

When dealing with high-dimensional data some preprocessing steps are advisable for logistic and computational reasons. Besides workload concerns, the preprocessing steps are key to ensure the good training performance of deep neural networks.

Resizing

As mentioned in the dataset description the videos were captured in 1640x1232 resolution, which is excessively high for this type of approach. Therefore, the images will be resized to a smaller dimension before being fed to the neural network. Knowing the first layer of the Inception V3 takes as input 299x299 images, that will be the target resolution.

Considering that in the future it is intended that the image acquisition system presented in chapter 2 will be capturing images and transmitting to a ground station, the video frames would have to be captured in an online scheme. To avoid further loss in image quality there was the need to find a solution for the video processing stage. FFMPEG was tested but did not yield satisfactory results, while VLC provided a much better solution in that regard. Being open-source software and cross-platform, VLC presents as a good solution for implementation on a ground-station for a online classification pipeline.

Since in this work the focus is on evaluating performance of models in offline classification, the frames were extracted from video sources that had been previously recorded. In VLC, using the scene filter the images were reduced to 299x299 with the original temporal resolution of the videos, which varies depending of the video source.

Input Normalization

The normalization of the input is a crucial step in deep neural networks to prevent exploding gradients. The Inception V3 model normalizes the input images to a [-2; 0] interval. This is done by dividing the input by 255, the maximum of a RGB channel , which normalizes the inputs to a 0 to 1 range. Subsequently, the inputs are subtracted by 1 and multiplied by two to yield the [-2; 0] range. The reference of the Inception V3 model [56] gives no indication of the reason behind choosing this particular normalization, but avoiding a zero mean associated with a [-1; 1] normalization, or the small values of a [0; 1] normalization are probably the underlying reasons. Since the feature part of the network is already optimized for this range, and that part will not be retrained, the same normalization is used.

Feature Extraction

To apply a transfer learning approach, the images of training dataset are run through the network to extract the feature representation of the penultimate layer. This is a pooling layer with a tensor output of 2048 dimension, which is usually referred to as the bottleneck layer. This procedure is performed once for every image, and the feature representations of the bottleneck layer are saved to an independent text file for each image. This process can take several hours depending on the size of the database, but the saved text files are read on future training sessions, which saves a lot of computational time in the long run. This approach also provides much flexibility, because if new images are added to the database later on, only their respective representations have to be extracted.

6.3 Training

6.3.1 Model Retraining

Having already extracted the features of the training dataset, the Inception V3 model was then retrained for the fire identification problem. To do so, additional operations were added to the TensorFlow graph of the model to perform the necessary computations. First, an input layer was created that sets the placeholders for the the bottleneck and ground truth tensor inputs. Second, a final training layer was added with the operations of the fully-connected layer. The outputs of the latter, will be the inputs to the cross-entropy scope and the train scope, which encompass the operations of the softmax classifier, the loss function and the gradient descent optimizer.

The Tensorboard representation of the operations described can be seen in further detail in figure 6.2. Note that the scopes represent the code implementation of the mathematical operations with TensorFlow, and the edges represent the data flow through the computational graph.

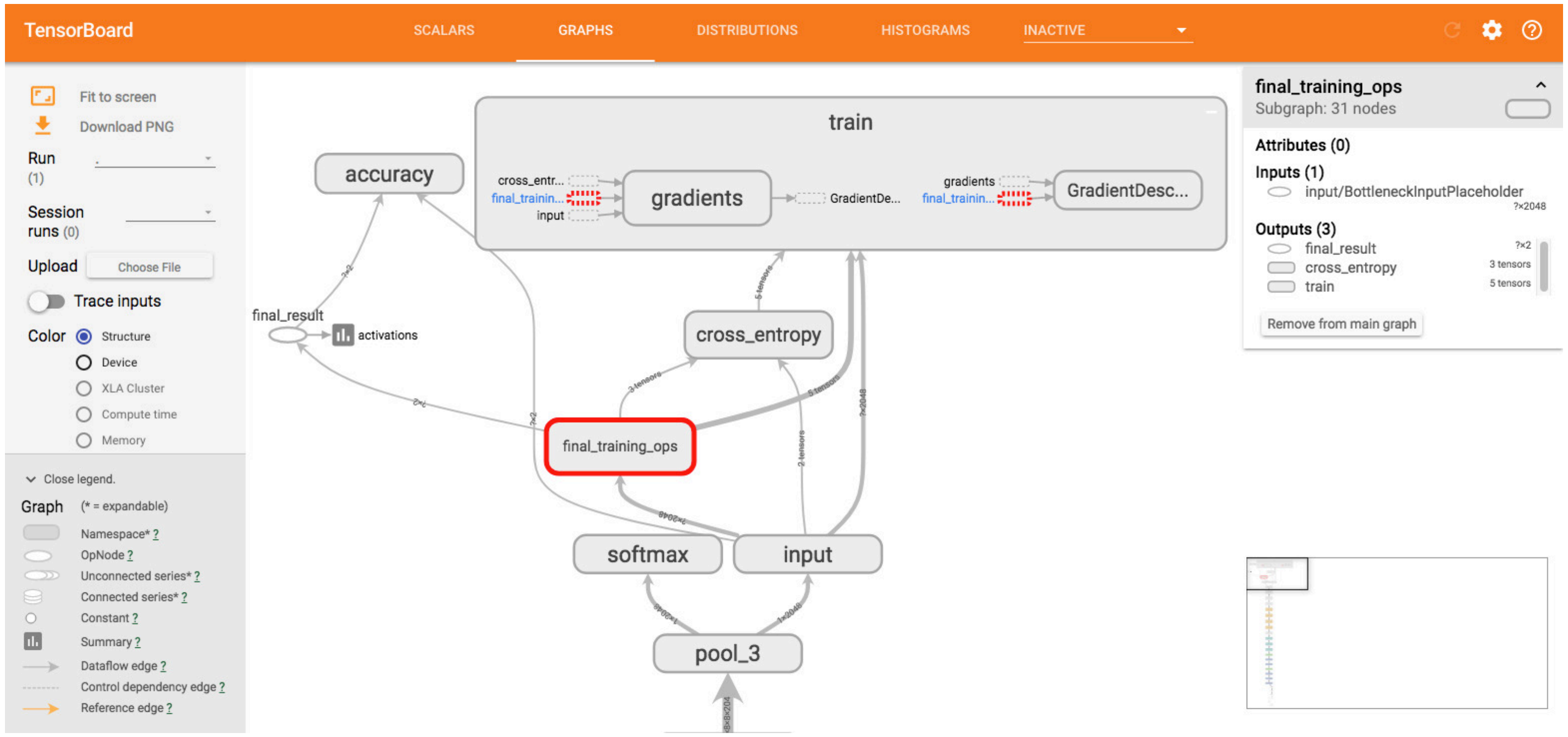


Figure 6.2: Tensorboard representation of the operations added to retrain Inception V3.

6.3.2 Hyperparameter Selection

To assess the best parameters for the model, several configurations were tested by changing the learning rate (LR), the training batch size ($Batch_{train}$) and the number of iterations (n_{iter}). High values of batch size were avoided since the memory cost would slow down the training process. If necessary, this would be compensated by increasing the number of iterations.

Since hyperparameter selection is an iterative process, a series of five distinct values of learning rate were tested: 0.005, 0.01, 0.05, 0.1, and 0.2. For these, training batch size was changed between 100 and 250, but it did not yield benefits in classification performance.

However, as figure 6.3 illustrates, the models achieve high performance on the first two runs through the data, which is due to the fact the dataset is composed by consecutive frames from few video sources. In this scenario, there is concern that these models will have limitations in generalization due to overfitting, but regardless this matter will be evaluated on the testing set.

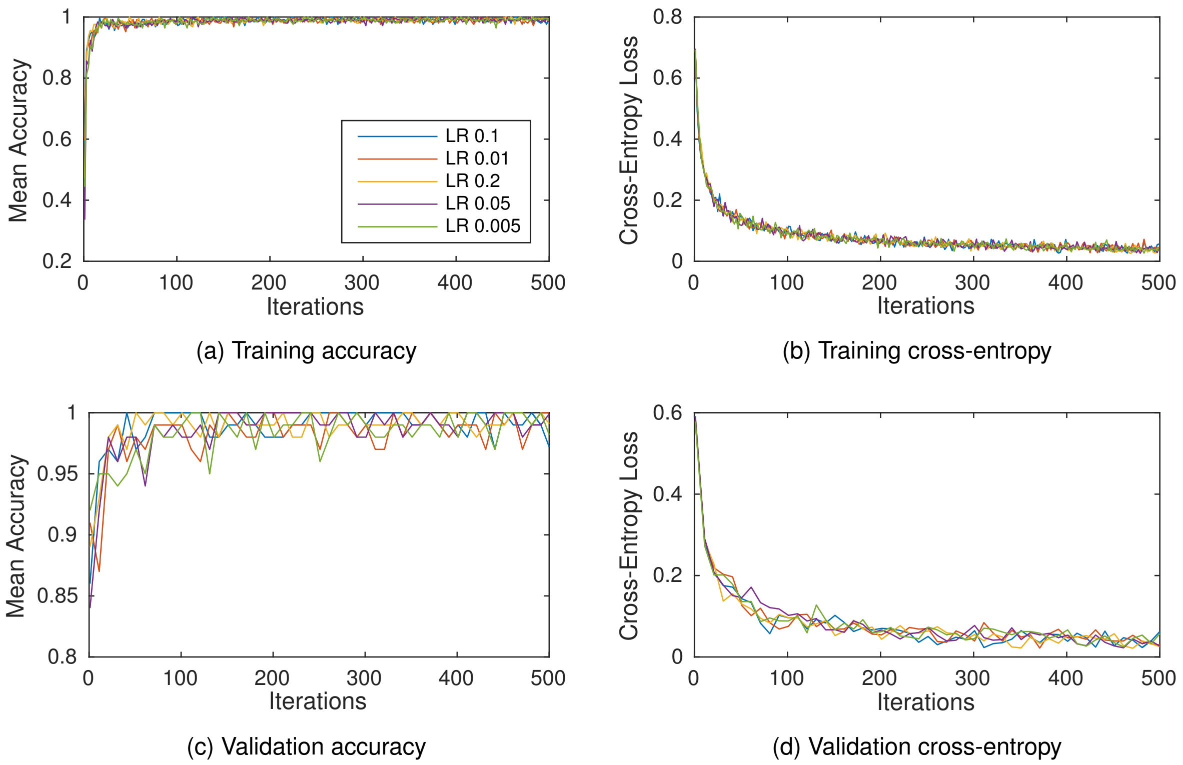


Figure 6.3: Comparison of models with different learning rate and batch size of 100.

As can be observed in the training and validation curves in figure 6.3 for accuracy and cross-entropy, the variation of the performance of the models with the change in learning rate is not significant. Furthermore, the convergence of the cross-entropy loss to small values in the training and validation sets demonstrates the training procedure was successful.

Due to the similarity in training performance of the models developed with a series of different hyperparameter configurations, the testing results are presented for a single model with the following hyperparameters: LR 0.01, $Batch_{train}$ 100, and n_{iter} 500.

6.4 Testing Results

In this section, to test the performance of the model, the situations analyzed are the same considered for the evaluation of the color segmentation heuristic, presented in chapter 4. Whereas that approach used thermal imaging, now the color images captured in those experimental trials are used to assess if an approach using this type of data would be effective.

As mentioned in section 6.1, this testing set is composed by three distinct cases, which will be studied individually in the following sections, with particular emphasis on the time elapsed between the fire ignition and detection. The overall performance for these test examples in terms of validity measures is discussed later in section 6.5.

6.4.1 Straw Example

For this example, images of straw burning were captured from an elevated platform in static conditions as was described in the experimental setup in section 2.2. A few selected frames from the video sequences obtained can be observed in figure 6.4.

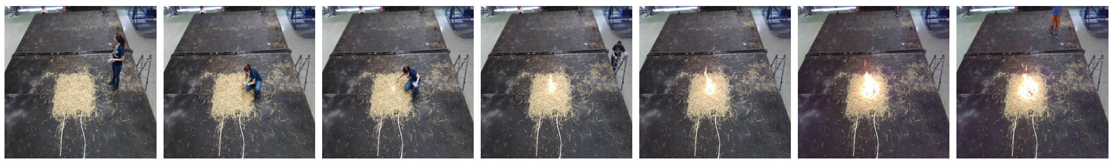


Figure 6.4: Color images from the straw burning trial.

The frames illustrate several points throughout the trial, as is visible by the gradual flame growth. However, it is important to notice, by zooming in on the third frame of figure 6.4, that when the fire is still at the beginning stage it is difficult to discern from the straw due to its small scale. While this frame should be classified as fire, it is reasonable to expect the network to face difficulties for these instances.

To analyze the performance of the network, the predicted scores of each frame are plotted in figure 6.5, along with the established ground truth values depicted in black.

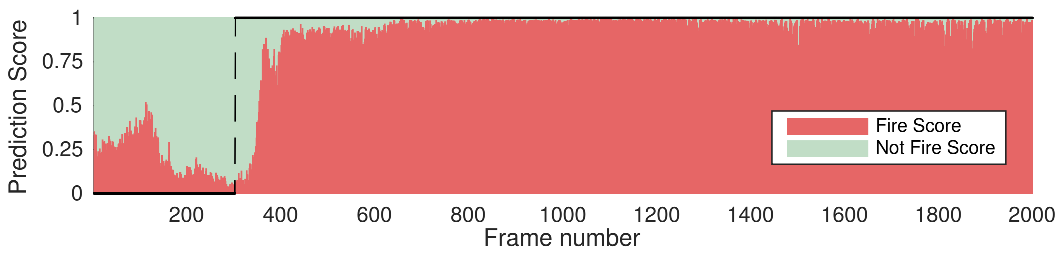


Figure 6.5: Straw burning (static): predicted scores and the ground truth over time.

Observing the output values, the detection is rapid and there are few misclassification prior to the fire. Considering the output of the network classifies a frame as fire when the predicted score of this class exceeds 0.5 and vice-versa, from frame 350 the network consistently detects fire with a high probability. Knowing the temporal resolution of this video source is 15 Hz, and the fire ground truth indicates fire in frame 300, the delay in detection is of approximately 3.3 seconds.

6.4.2 Tent Example

The second example concerns the burning of a camping tent, where the fire source is inside it. While in early stages of this trial there are small light flashes caused by the fire, the flame is only visible in color images when it starts burning the exterior of the tent. Therefore, the ground truth for this example was established significantly later than the actual start of the fire. Notice, by observing figure 6.6, that this situation is considerably different than the burning tent examples used for training, depicted in figure 6.1. Hence, this example provides a good test for the generalization of the model.

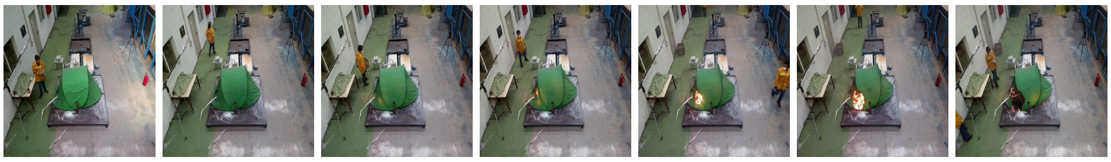


Figure 6.6: Color images from the tent burning trial.

The output of the retrained model is depicted in figure 6.7, as well as the ground truth for each frame represented by the black line. Once again, the latter takes the value of 1 when the fire is identified and 0 when it is not. For the tent case, the ground truth was defined conservatively, and is only when the fire is visible at naked-eye that it assumes the value of one.

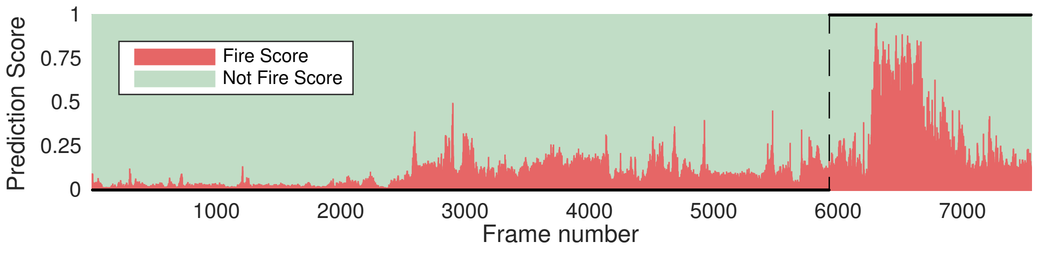


Figure 6.7: Tent burning: predicted scores and the ground truth over time.

Comparing with the point where fire should be detected for the first time, at frame 5927, the network identifies the fire with a delay of 360 frames, which means the first alarm would be given in 24 seconds.

While the network was not trained to detect fire in a situations where is not directly visible, it is noticeable that the prediction score of the fire class starts increasing well before fire is detected. Taking into account, that in reality the fire inside the tent starts around frame 250, the output for this case demonstrates the delay in detection would be much longer resorting to color images, as a result of the limitations in sensing capability associated with this type of camera.

Moreover, the reduction of the prediction score after frame 6700 is due to the fact that fire is at a very small scale in those respective frames. This can be verified by zooming in on the last frame of figure 6.6. This finding highlights an important issue that has to be taken into consideration for an application relying on an aerial platform. The limitations of this type of models will have to be further investigated in this regard, to evaluate the tradeoff between area coverage and the pixel resolution necessary for fire identification.

6.4.3 Drone Example

The straw burning example captured from a moving platform was the test where the color segmentation heuristic showed most limitations, so there is increased interest in investigating if this approach works better for this case. Furthermore, having a moving platform provides flexibility in area coverage over a fixed platform, so it is important to evaluate if this is a feasible option.

For this example the image sequence has considerable more variance than the example given in section 6.4.1. Now, as figure 6.8 depicts, the point of view changes in every frame due to the movement of the drone.

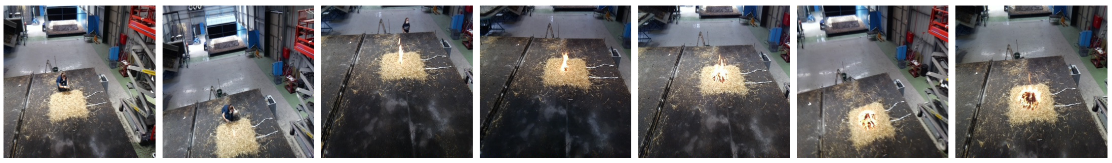


Figure 6.8: Color images from the straw burning trial captured from a drone.

Since the camera is forced to refocus constantly, the brightness and contrast in the frames vary as well. Additionally, now the frames have more noise and blurring which adds more difficulty to the classification task.

Akin to the previous examples, to evaluate the performance for this test case, the output of the model is juxtaposed with the ground truth values in figure 6.9.

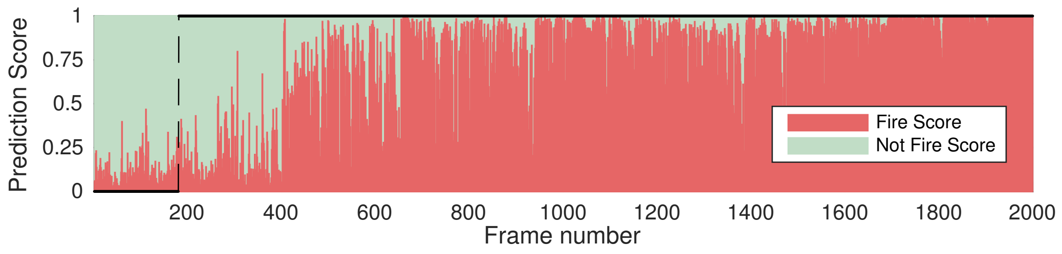


Figure 6.9: Straw burning (drone): predicted scores and the ground truth over time.

Observing the output of the network in figure 6.9, there is increased stochastic behavior which would be to expect in this type of case. Despite the presence of noise, and some delay in detecting the fire, from around frame 400 the vast majority of the frames are correctly classified.

The ground truth value indicates that the fire starts at frame 180, and the first fire alarm would occur 75 frames later. Hence, for this case, the delay is of 5 seconds, which is a very promising result. Considering that for this trial there were no stabilization systems for the camera, the performance is likely to improve if those type of systems are employed.

Since convolutional neural networks provide translation invariance, the movement of the image acquisition platform does not affect this approach as much as the color segmentation heuristic. The main problem with the latter in this case was that the movement of the drone caused the image statistics to vary in a highly stochastic manner. Additionally, unlike the previous approach which relied on the sensor response over time, the output of the neural network does not have any temporal dependencies and classifies each image independently of the previous.

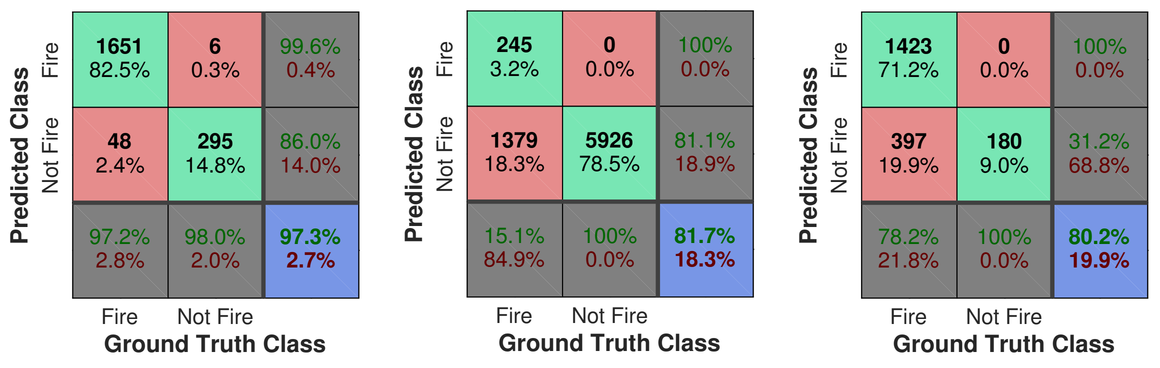
6.5 Discussion

The previous sections presented the image classification results for the retrained network for three distinct test cases. So far, the analysis focused on the evolution of the prediction scores over time. However, the results of the model have yet to be evaluated using a validity measures approach, which will be analyzed in section 6.5.1. Subsequently, having analyzed the response of the DNN model for each of the tests following the point of ignition methodology, now it is important to frame these results in context with the ones from chapter 4. This allows for a clear understanding of the strengths and limitations of the different sensors of the image acquisition system presented in chapter 2.

6.5.1 Performance Analysis

In what concerns the validity measures approach, it is important to keep in perspective this work is not a benchmarking task. Hence, from a variety of models developed, due to their similarity in performance, this work only presents the results for one model, which is evaluated for the testing cases studied.

In that sense, figure 6.10 summarizes the performance measures for each test example in the form of a confusion matrix for the selected model.



(a) Straw burning (static) - Test 1 (b) Tent burning (static) - Test 2 (c) Straw burning (drone) - Test 3

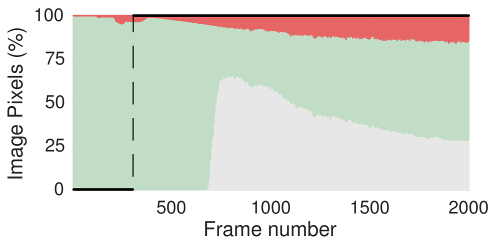
Figure 6.10: Performance evaluation of the testing set.

The results demonstrate the model achieves a high accuracy value for the easier of the test cases (fig. 6.10a), but there is still room for improvement in the tent and drone cases, (fig. 6.10b and fig. 6.10c). Focusing on the case of figure 6.10b, there is a 18.3% of misclassifications due to false negatives. This is too high considering the true positives represent only 3.2% of the batch for this test, and makes the sensitivity of the model of only 15%. The reason for this is the small scale of the fire in many of the frames. This rate would be worrisome if a significant percentage of frames with large flames was being misclassified, which is not the case. For the test depicted in figure 6.10c there is a 19.9% of false negatives. While this is undesirable, taking into account the increased noise of image acquisition from a moving platform, the drop in accuracy is understandable. This result reveals this approach has great potential for application in moving platforms, not only limited to fire detection, but other computer vision applications. The reduction of the noise in the frames would mitigate the incidence of false negatives.

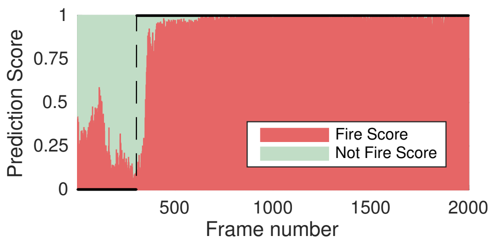
However, it should be noted that these sets are composed of consecutive video frames, which means that the content of the images has a high degree of similarity. This fact should be taken into consideration when assessing the performance of the model. Furthermore, since the training dataset is composed of a small number of situations, the generalization of the model will be limited. Nevertheless, considering the situations that make up the testing dataset were developed in a similar scenario, these serve to demonstrate the feasibility of employing deep learning approach to this problem.

6.5.2 Comparison of Results

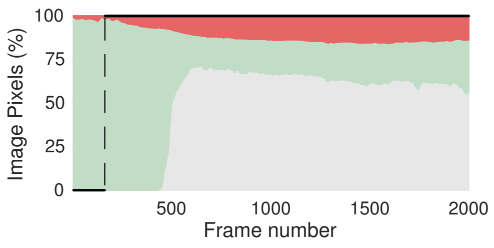
In figure 6.11, the results from the color segmentation heuristic, presented in chapter 4, are juxtaposed with the ones from the image classification, presented in this chapter. From these, the central focus is on the analysis of the images captured with the system proposed in chapter 2, thus the radiometric thermal images are excluded. However, it is important to bring attention to a few distinctions between the two approaches.



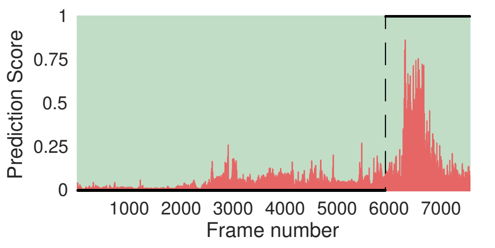
(a) Nonradiometric: Straw burning (static)



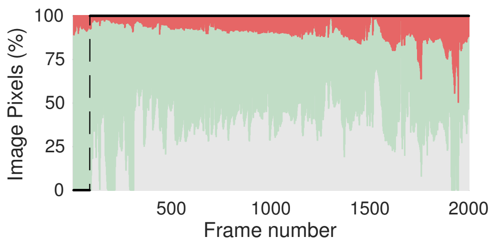
(b) Inception: Straw burning (static)



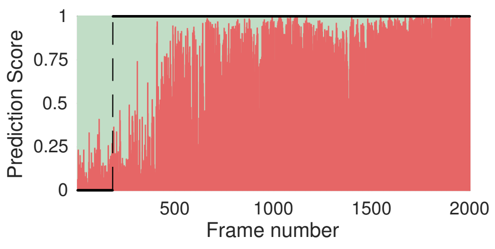
(c) Nonradiometric: Tent burning (static)



(d) Inception: Tent burning (static)



(e) Nonradiometric: Straw burning (drone)



(f) Inception: Straw burning (drone)

Figure 6.11: Comparison of the segmentation heuristic for thermal images (left) versus the Inception classification for color images (right).

Recalling that part of the first approach was applied to thermal images captured with a nonradiometric camera, the proposed heuristic allowed for the segmentation of three classes of colors: gray, green, and red. One of the discoveries from chapter 4, was that the trigger for an alarm should be set by monitoring the percentage of gray and green pixels rather than the red, as was previously thought. Moreover, observing the test cases on the left side of figure 6.11, the threshold of an alarm based on the percentage of gray pixels would have to be considerably low to minimize the time between ignition and detection.

In turn, as we have seen throughout this chapter, the second approach concerns the image classification of color images without prior feature extraction. The results from this approach are compiled again to facilitate the comparison, and are featured in the right side of figure 6.11.

First, comparing the results for the straw burning example, the approach using thermal data, illustrated in fig. 6.11a, exhibits more limitations than expected in a situation where fire is directly visible. On the contrary, the graph in fig. 6.11b demonstrates that the classification using the Inception V3 model is faster to detect the fire using color images, and exhibits high consistency it its predictions over time.

While this happens for the first test case, for the second, depicted in figures 6.9c and 6.9d, the opposite occurs. These results concern the burning tent, and the point of ignition is detected earlier using the thermal sensor. Note the difference between the established ground truth values for each case differs in 5767 frames, approximately 6 minutes and 40 seconds. In color images, the fire can only be detected when it is visible at naked-eye, thus using this type of data the detection happens later. Despite a slight delay in the response of the thermal sensor, this example demonstrates the value a thermal camera adds for an earlier fire detection application. For this case an alarm would be triggered using thermal data approximately 6.5 minutes earlier than if resorting to the color images.

Regarding the last row of figure 6.11, which compares the results for the straw burning captured from a drone, several conclusions can be drawn. First, the stochastic response in both approaches increases the difficulty in classification considerably. On the one hand, observing figure 6.11e relative to the segmentation of the thermal images, the threshold for an alarm would have to be set very low to capture the fire occurrences consistently. However, considering the low saturation level of this sensor identified in chapter 3, relying solely on this sensor would lead to many false alarms. On the other hand, despite some misclassification due to noise the image classification using color images (fig. 6.11f) yielded very promising results. These results attest to the feasibility of using aerial platforms for fire detection applications.

To conclude this overview of the results it matters to quantify for both methodologies the time elapsed from the ignition to the first instance of fire detection. Setting a threshold value of 0.5 (50%), the results for each test are summarized in table 6.2.

Table 6.2: Time intervals for the first fire alarm for the experimental tests.

Data Type	Test 1	Test 2	Test 3
Thermal Nonradiometric	28 s	22.6 s	2.8 s
Color	3.4 s	6m48s	5 s

Overall, despite the strengths and limitations of each approach, from the point of ignition to the first fire alarm, the detection is achieved in short timeframes for both approaches. Therefore, a framework for early fire detection would benefit from integrating these two types of sensory data into a single algorithm, in order develop a more general solution.

Chapter 7

Conclusions

Lately, due to climate change, conditions of high ignition propensity are more frequent, which results in an increase in fire incidents. These are escalating in severity, which prompts the necessity of mitigating its devastating consequences. Currently, there are several projects implemented with the objective of reducing fire propagation by early detection, yet their success is still limited.

The present work was developed under the scope of project Firecamp 2, which centers on studying fire risk in areas dedicated to camping and caravanning activities. The investigation scope of this project is not limited to camping parks, and is extensible to fire risk situations in the wildland-urban interface. In this context, this work centered on analyzing different imaging sensory data and exploring computer vision techniques for detection of fire incidents in an initial stage.

To address a problem of this complexity, for which the available data was very limited, the first step required the development of experimental tests for image acquisition purposes. In order to perform realistic tests, a data acquisition system was designed to be integrated on a mobile aerial platform e.g. a drone, a balloon. While there was the intention of performing tests in an outdoor setting, due to the early arrival of fire season in 2017, for safety reasons, these tests to be conducted in June, were only carried out in indoor laboratory conditions, at the Laboratory of Forestry Fire Studies in Lousã.

One of the principal objectives of this work was to evaluate the adequacy of different imaging sensors for this task. To that end, an image acquisition system was designed with two different cameras to capture aerial images in the thermal and visible range. Since the image acquisition was performed synchronously, the tests resulted in a new dataset for fire detection composed by thermal nonradiometric and color images.

In addition to those images, the experimental trials were also recorded with a thermal radiometric camera. The access to radiometric and nonradiometric data for the same situations allowed for the development of a deep study of the characteristics of these sensors and their differences. One of the key findings of this analysis was the discovery of the thermal sensing ranges of both cameras. It was uncovered that for the nonradiometric camera, FLIR Vue Pro, the sensor upper saturation limit is of 150^{o}C. While such a low saturation level is not ideal for a fire detection problem, this information allows for a better interpretation of the image data provided by this camera. Moreover, it was identified that using this filter the camera adjusts the color scale according to the highest temperature in the scene.

Having now different types of data to explore, various avenues of research were taken to evaluate its merits and limitations. A first approach centered on thermal images and the development of a color segmentation heuristic for fire detection. Since the images from the thermal cameras are encoded in pseudo-color, the central advancement revolved on uncovering the color palette applied by the camera firmware of FLIR Vue Pro. The *GrayRed* color filter was identified and subsequently applied to the radiometric data captured by FLIR SC660, in order to compare the response of these two sensors.

By devising a color segmentation heuristic the color palette was partitioned into three different classes: gray, green, and red. The analysis of the variation of these classes provided insight into the response of the sensor in terms of RGB encoding. While there was a prior misconception that fire could be identified by monitoring the percentage of red pixels, the results demonstrated that for this application it is preferable to monitor the percentage of gray and green pixels. This is the case because when there are large temperature differences in the field of view of the camera, the variation of gray and green pixels is almost inversely proportional.

While this study enriched the understanding of the inner workings of the thermal cameras, the analysis of the data from various experimental tests showcased the limitations of this type of approach. Due to excessive variability of the color levels, establishing an algorithm based on these three variables would have strong limitations in generalization for new scenarios. However, if a solution is required for fire detection for very specific contexts, a color segmentation heuristic could be fine-tuned to target a limited array of cases, by establishing a rule-based algorithm.

The second approach to address the fire detection problem explored the color imaging data, without prior feature extraction. This was achieved by resorting to a state-of-the-art deep neural network, i.e. the Inception V3 model, and applying a transfer learning scheme. Since this type of approach requires large amounts of data, the previous color image dataset was augmented with RGB images from additional tests. Although only offline classification was performed, the preprocessing procedure to extract frames from the video sequences was established having an online classification pipeline in mind.

In what concerns the retraining of Inception V3, only the classifier part of the model was retrained using TensorFlow. This deep learning library provided a fast training framework, and offers an efficient platform for the future implementation of online classification in real-time.

Regarding the classification results, this approach demonstrated high accuracy for the cases where the fire is directly visible. From the results of this approach the test performed with images acquired from a drone, has special importance. Deep learning approaches are mostly applied in offline tasks with large datasets and to image data acquired from static conditions. It is only very recently that these models achieved the computational efficiency to be applied to more flexible platforms used in mobile robotics, with solutions such as MobileNets [59]. Therefore, the results from the test with the drone demonstrates these techniques can be successfully applied to the fire detection problem and possibly other computer vision tasks.

Although the two approaches proposed to this problem are very distinct in nature, the results provided a clear understanding of the limitations of each and the benefits using both can bring. Despite the fact that the tests presented in this work contemplate a limited set of situations, they demonstrated the value of designing a system that can capture different types of data. While the thermal imaging approach extends the sensing ability enabling faster detection, the deep learning approach with color images has the potential to provide more general and robust algorithms. Overall, the comparative analysis between the two approaches taken demonstrated that for a fire detection application the development of algorithms should encompass thermal and color image data.

Granted that deep learning models have achieved state-of-the-art results in a variety of close-set problems, the application of these techniques to open-set problems is still in early stages. The limited amount of data for this problem was one of the main difficulties faced during the development of this work, and is a common hurdle in applying deep learning approaches to real problems. In that sense, given the current relevance of this topic, the open release of the acquired dataset provides a basis for the continuation of this work, and opens up new opportunities for the development of different approaches to this problem.

Future Work

Given the current pace of innovation in the field of deep learning there is great potential for further research using this approach. To target this problem, the following topics are suggested for the continuation of this work:

- Extend image classification to video classification using recurrent neural networks i.e. Long-Short-Term-Memory networks in conjunction with CNNs, in order to study spatiotemporal features.
- Development of hybrid learning networks using thermal and color image data in tandem.
- Application of deep learning models for real-time fire identification.

Bibliography

[1] J. San-Miguel-Ayanz, T. Durrant, R. Boca, G. Libertà, F. Boccacci, M. Di Leo, J. López Pérez, and E. Schulte. Forest Fires in Europe, Middle East and North Africa 2015. Technical report, Joint Research Centre - European Commission, 2016.

[2] E. Chuvieco. *Earth Observation of Wildland Fires in Mediterranean Ecosystems*. Springer-Verlag Berlin Heidelberg, 2009. ISBN 3642017533,9783642017537.

[3] J. Keeley, W. Bond, R. Bradstock, J. G. Pausas, and P. Rundel. *Fire in Mediterranean Ecosystems. Ecology, evolution and management*. Cambridge University Press, 2012. ISBN 9780521824910.

[4] M. V. Moreno and E. Chuvieco. Characterising fire regimes in spain from fire statistics. *International Journal of Wildland Fire*, 22(3):296–305, 2013. doi: 10.1071/WF12061.

[5] M. Salis, A. A. Ager, M. A. Finney, B. Arca, and D. Spano. Analyzing spatiotemporal changes in wildfire regime and exposure across a mediterranean fire-prone area. *Natural Hazards*, 71(3): 1389–1418, Nov. 2013.

[6] A. A. Ager, N. M. Vaillant, and M. A. Finney. Integrating fire behavior models and geospatial analysis for wildland fire risk assessment and fuel management planning. *Journal of Combustion*, 2011(1): 1–19, 2011.

[7] E. Chuvieco, I. Aguado, S. Jurdao, M. L. Pettinari, M. Yebra, J. Salas, S. Hantson, J. de la Riva, P. Ibarra, M. Rodrigues, M. Echeverría, D. Azqueta, M. V. Román, A. Bastarrika, S. Martínez, C. Recondo, E. Zapico, and F. J. Martínez-Vega. Integrating geospatial information into fire risk assessment. *International Journal of Wildland Fire*, 23(5):606, 2014. doi: 10.1071/WF12052.

[8] S. Marques, J. G. Borges, J. Garcia-Gonzalo, F. Moreira, J. M. B. Carreiras, M. M. Oliveira, A. Cantarinha, B. Botequim, and J. M. C. Pereira. Characterization of wildfires in Portugal. *European Journal of Forest Research*, 130(5):775–784, Jan. 2011.

[9] S. L. J. Oliveira, J. M. C. Pereira, and J. M. B. Carreiras. Fire frequency analysis in Portugal (1975 - 2005), using Landsat-based burnt area maps. *International Journal of Wildland Fire*, 21(1):48–13, 2012.

[10] L. A. Arroyo, C. Pascual, and J. A. Manzanera. Fire models and methods to map fuel types: The role of remote sensing. *Forest Ecology and Management*, 256(6):1239–1252, Sept. 2008.

[11] D. E. Calkin, M. P. Thompson, M. A. Finney, and K. D. Hyde. A real-time risk assessment tool supporting wildland fire decisionmaking. *Journal of Forestry*, 109(5):274–280, 2011.

[12] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou. An early fire-detection method based on image processing. In *2004 International Conference on Image Processing*. IEEE, 2004.

[13] T. Çelik and H. Demirel. Fire detection in video sequences using a generic color model. *Fire Safety Journal*, 44(2):147–158, Feb. 2009.

[14] W.-B. Horng, J.-W. Peng, and C.-Y. Chen. A new image-based real-time flame detection method using color analysis. In *Proceedings. 2005 IEEE Networking, Sensing and Control.* IEEE, 2005.

[15] G. Marbach, M. Loepfe, and T. Brupbacher. An image processing technique for fire detection in video images. *Fire Safety Journal*, 41(4):285–289, June 2006.

[16] B. U. Töreyin, Y. Dedeoğlu, U. Güdükbay, and A. E. Çetin. Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49–58, Jan. 2006.

[17] B. U. Töreyin. Fire detection in infrared video using wavelet analysis. *Optical Engineering*, 46(6): 067204, June 2007.

[18] I. Bosch, S. Gomez, R. Molina, and R. Miralles. Object discrimination by infrared image processing. In *Lecture Notes in Computer Science*.

[19] S. Verstockt, A. Vanoosthuyse, S. V. Hoecke, P. Lambert, and R. V. de Walle. Multi-sensor fire detection by fusing visual and non-visual flame features. In *Lecture Notes in Computer Science*, pages 333–341. Springer Berlin Heidelberg, 2010.

[20] C. R. Steffens, R. N. Rodrigues, and S. S. da Costa Botelho. *Non-stationary VFD Evaluation Kit: Dataset and Metrics to Fuel Video-Based Fire Detection Development*. Springer International Publishing, 2016.

[21] A. E. Çetin, K. Dimitropoulos, B. Gouverneur, N. Grammalidis, O. Günay, Y. H. Habiboğlu, B. U. Töreyin, and S. Verstockt. Video fire detection – review. *Digital Signal Processing*, 23(6):1827–1843, Dec. 2013.

[22] T. ho Chen, Y. hui Yin, S. feng Huang, and Y. ting Ye. The smoke detection for early fire-alarming system base on video processing. In *2006 International Conference on Intelligent Information Hiding and Multimedia*. IEEE, dec 2006.

[23] B. C. Ko, K.-H. Cheong, and J.-Y. Nam. Fire detection based on vision sensor and support vector machines. *Fire Safety Journal*, 44(3):322–329, Apr. 2009.

[24] T. X. Tung and J.-M. Kim. An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems. *Fire Safety Journal*, 46(5):276–282, July 2011.

[25] S. P. Hohberg. Wildfire smoke detection using convolutional neural networks. Master's thesis, Simon Philipp Hohberg, 2015.

[26] Z. Wu, T. Yao, Y. Fu, and Y. Jiang. Deep learning for video classification and captioning. *CoRR*, abs/1609.06782, 2016.

[27] M. Almeida, J. R. Azinheira, J. Barata, K. Bousson, R. Ervilha, M. Martins, A. Moutinho, J. C. Pereira, J. C. Pinto, L. M. Ribeiro, J. Silva, and D. X. Viegas. Analysis of fire hazard in campsite areas. *Fire Technology*, 53(2):553–575, Apr. 2016.

[28] *FLIR Vue Pro and Vue Pro R User Guide*. FLIR, July 2016. Version 110.

[29] W. Minkina and S. Dudzik. *Infrared Thermography: Errors and Uncertainties*. J. Wiley, 2009. ISBN 9780470747186.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[31] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012. ISSN 1053-5888.

[32] D. Shapiro. NVIDIA Takes AI From Research to Production in New Work with Volvo, VW, ZF, Autoliv, HELLA. URL `https://blogs.nvidia.com/blog/2017/06/27/nvidia-european-automotive-ai/`. Accessed: 2017-09-25.

[33] NVIDIA. Partner innovation: Accelerating automotive breakthroughs. URL `http://www.nvidia.com/object/automotive-partner-innovation.html`. Accessed: 2017-09-25.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[35] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using Convolutional Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. doi: 10.1109/cvpr.2015.7298664. URL `https://doi.org/10.1109/cvpr.2015.7298664`.

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[38] P. Langley and H. A. Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38:55–64, 1995.

[39] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[40] F. Rosenblatt. The perceptron, a perceiving and recognizing automaton (project para). Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[41] D. Cox and T. Dean. Neural networks and neuroscience-inspired computer vision. *Current Biology*, 24(18):R921 – R929, 2014.

[42] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[48] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

[49] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. *CoRR*, abs/1605.07678, 2016. URL `http://arxiv.org/abs/1605.07678`.

[50] R. Collobert, S. Bengio, and J. Marithoz. Torch: A modular machine learning software library, 2002.

[51] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[52] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[53] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens,

B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[54] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014. URL `http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf`.

[55] TensorFlow Development Team. Inception in TensorFlow, Github Repository. URL `https://github.com/tensorflow/models/tree/master/research/inception`. Accessed: 2017-09-25.

[56] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *CoRR*, abs/1512.00567, 2015.

[57] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[58] P. Sadowski. Notes on backpropagation. URL `https://www.ics.uci.edu/~pjsadows/notes.pdf`. Accessed: 2017-09-25.

[59] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.