Ministry of Higher Education and Scientific Research
University of Abbes Laghrour Khenchela
Faculty of Science and Technology

Department of Mathematics and Computer Science

Serial N°: .....

Graduation Thesis

For Master degree in Computer Science

Speciality : Security and web technology

Deep learning for monitoring forest fires, Aïn Mimoune forest application

Submitted By:

- Oucif Abdelkarim
- Belaabed Miloud

Supervised By:

Dr. Abbas Fayçal

President :

Supervisor : Dr. Abbas Fayçal.

Examiner :

2021/2022


بسم الله الرحمن الرحيم
١٤٣٨

Dedicate

“

With a feeling full of love, sincerity, and fidelity, I dedicate this humble work:
TO the man of my life, my eternal example, my moral support, and source of joy and happiness, the one who has always sacrificed himself to see me succeed, May Allah cure you, to you My father (Bachir).

TO the light of my days, the source of my efforts, the flame of my heart, my life, and my happiness; mom that I adore (Dalila).

TO my brother Chaban and my sister, for all the caring moments, for the warmth and true happy moments.

TO the treasure of me live, and light of dark nights, the person who gave all the support and true care, my brother "Miloud belaabed", for the hard work and efforts, Without which this work would not have been completed,the true friendship "There Are Big Ships and Small Ships. But The Best Ship Of All Is Friendship".

TO every person, every clean heart, every person that helped us even with the smallest word. I would never forget it, it's all in my heart.

”

Karim

“

With a feeling full of love, sincerity, and fidelity, I dedicate this humble work:
TO the man of my life, my eternal example, my moral support, and source of joy and happiness, the one who has always sacrificed himself to see me succeed, May Allah protect you, to you My father (Ismail).

TO the light of my days, the source of my efforts, the flame of my heart, my life, and my happiness; mom that I adore (Mesaouda).

TO all of my brothers and my sister, for all the caring moments, for the warmth and true happy moments.

TO the treasure of me live, and light of dark nights, the person who gave all the support and true care, my brother "Oucif Abdelkarim", for the hard work and efforts, Without which this work would not have been completed, the true friendship "One loyal friend is worth more than a thousand fake ones".

TO every person, every clean heart, every person that helped us even with the smallest word. I would never forget it, it's all in my heart.

”

Miloud

Remerciements

First of all, we would like to thank Allah the Almighty and Merciful, who gave us the courage during these long years of study, the strength and the patience to accomplish this modest work.

We would like to sincerely thank our supervisor, Mr: Abbes Fayçal, for his valuable advice, guidance, trust and patience which constituted a considerable contribution without which this work could not have been carried out successfully. May he find in this work a living tribute to his high personality.

These thanks go first of all to the faculty and administrative staff of the Faculty of Science and Technology, then to all the professors who taught us and who, through their skills, supported us in the pursuit of our studies.

Our heartfelt thanks also go to the members of the jury for their interest in our research.

We thank our parents with all our hearts, for always supporting us and believing in us. To our families and friends who through their prayers and encouragement, we were able to overcome all the obstacles.

Finally, we would also like to thank all the people who participated directly or indirectly in the realization of this work.

Thanks to everyone

Abstract

In recent years, major fires with devastating consequences have occurred in various regions of Algeria. The region of the wilaya of Khenchela was affected by heat waves between July and August which caused major fires in several forests, precisely in the region of Ain Mimoune. This thesis addresses the problem of fire and smoke detection in videos, with the aim of early detection of fires in the forests of Aïn Mimoune.

As we know current methods use traditional image processing methods that rely primarily on human-made extracted features, which aren't always appropriate to all forest environments.

To solve this issue, we have proposed a custom model based on deep learning techniques that helps in early fire and smoke detectors, the final proposed model integrates two independent models. The first one uses the features extraction power of VGG16 to avoid false positives, while the second one uses YOLOV5 that highlight fire or smoke with a surrounding bounding box.

The proposed solution takes into consideration the time and GPU resources issues, so it could be applicable in real-world scenarios.

To test our proposed solution on a forest-similar environment, we have created a forest-simulation environment using the Unity3D game engine, which gave us good results even with a lack of GPU resources, this step has opened a big door for future perspectives and suggestions that can be done concerning the fire detection problem.

In the end, to make our solution public and available to everyone, we have used the Django framework to create a web application interface that uses the final proposed algorithm.

Keywords : Deep learning, Ensemble learning, Transfer learning, Fine-tuning, 3D simulation, Web application

---

Résumé

Ces dernières années, des incendies importants aux conséquences dévastatrices se sont produits dans diverses régions d'Algérie. La région de la wilaya de Khenchela a été touchée par des vagues de chaleur entre juillet et août qui ont provoqué des incendies importants dans plusieurs forêts, précisément dans la région d'Aïn Mimoune. Cette thèse aborde le problème de la détection de feu et de fumée en vidéo, dans le but de détecter précocement les incendies dans les forêts d'Aïn Mimoune.

Comme nous le savons, les méthodes actuelles utilisent des méthodes de traitement d'image traditionnelle qui repose principalement sur des caractéristiques extraites par l'homme, qui ne sont pas toujours appropriées à tous les environnements forestiers.

Pour résoudre ce problème, nous avons proposé un modèle personnalisé basé sur des techniques d'apprentissage profond qui aide à la détection précoce des incendies et de la fumée, le modèle final proposé intègre deux modèles indépendants. Le premier utilise la puissance d'extraction des caractéristiques de VGG16 pour éviter les faux positifs, tandis que le second utilise YOLOV5 qui met en évidence le feu ou la fumée avec une boîte de délimitation environnante.

La solution proposée prend en compte les problèmes de temps et de ressources GPU, elle pourrait donc être applicable dans des scénarios du monde réel.

Pour tester notre solution proposée sur un environnement similaire à la forêt, nous avons créé un environnement de simulation forestière en utilisant le moteur de jeu Unity3D, qui nous a donné de bons résultats même avec un manque de ressources GPU, cette étape a ouvert la porte à de futures perspectives et suggestions concernant le problème de la détection des incendies.

En fin de compte, pour rendre notre solution publique et accessible à tous, nous avons utilisé le framework Django pour créer une interface d'application Web qui utilise l'algorithme final proposé.

Mots clés : Apprentissage profond, Apprentissage d'ensemble, Transfert learning, Fine tuning, Simulation 3D, Application Web

---

ملخص

في السنوات الأخيرة ، اندلعت حرائق كبيرة ذات عواقب وخيمة في مناطق مختلفة من الجزائر. حيث تأثرت منطقة ولاية خنشلة بموجات الحر بين شهري يوليو وأغسطس ، مما تسبب في حرائق كبيرة في عدة غابات ، وتحديداً في منطقة عين ميمون. نتناول في هذه الرسالة مشكلة كشف الحرائق والدخان بالفيديوهات بهدف الكشف المبكر عن الحرائق في غابات عين ميمون.

و كما نعلم ، تستخدم الأساليب الحالية طرق معالجة الصور التقليدية التي تعتمد بشكل أساسي على الميزات المستخرجة من طرف الإنسان ، والتي لا تتناسب دائمًا مع جميع بيئات الغابات.

لحل هذه المشكلة ، اقترحنا نموذجًا مخصصًا يعتمد على تقنيات التعلم العميق التي تساعد في الكشف المبكر عن الحرائق والدخان ، النموذج النهائي المقترح يدمج نموذجين مستقلين. الأول يستخدم قوة استخراج الميزات لـ VGG16 لتجنب التنبؤات الخاطئة ، بينما يستخدم الثاني YOLOV5 الذي يقوم بتحديد النار أو الدخان بصندوق محيط.
يأخذ الحل المقترح في الاعتبار مشاكل الوقت وموارد وحدة معالجة الرسومات، لذلك يمكن أن يكون قابلاً للتطبيق في سيناريوهات العالم الحقيقي.يوهات العالم الحقيقي.

لاختبار الحل المقترح في بيئة مشابهة للغابات، أنشأنا بيئة محاكاة للغابات باستخدام محرك اللعبة ،Unity3D والذي أعطانا نتائج جيدة حتى مع نقص موارد وحدة معالجة الرسومات، فتحت هذه الخطوة بابًا كبيرًا لوجهات النظر والاقتراحات المستقبلية التي يمكن القيام بها فيما يتعلق بمشكلة الكشف عن الحرائق.

في النهاية، لجعل حلنا عامًا ومتاحًا للجميع، استخدمنا إطار عمل Django لإنشاء واجهة تطبيق ويب تستخدم الخوارزمية النهائية المقترحة.

كلمات مفتاحية : التعلم العميق ، التعلم الجمعي ، نقل التعلم ، الضبط الدقيق, محاكاة ثلاثية الأبعاد، تطبيق الويب

Table of contents

List of Figures

List of Tables

List of Algorithms

General Introduction

In recent years, major fires, with devastating consequences, have occurred in various regions of Algeria where heat waves caused major fires in several forests between July and August which not only cause serious economic losses and destroy the ecological environment but also pose a great threat to the safety of human life. Forest fires usually spread quickly and are difficult to control in a short time.

Many traditional solutions deal with fire detection using many approaches, from image processing using color spaces (especially HSV) to the sensors-based solutions that monitor the fire.

However, using these classic approaches will present huge limitations from performance to time and resources required to implement these solutions, as an example due to the limitation of the detection range of sensors, the monitoring system can not cover a wide range of monitoring areas, and the traditional detection methods can not give valuable information of detected fires, and the color based system won't be that intelligent to distinguish between fire and orange cloths as an example.

Recently, with the popularization of intelligent monitoring equipment, the development of image processing technology, deep learning, and intelligent optimization algorithms, the fire monitoring problem based on video analysis has attracted more and more attention from researchers with the help of security monitoring, video monitoring, and other technologies. Based on fire detection. It is a low-cost, efficient fire detection system that can significantly reduce casualties and property damage caused by fires.

This led us, to propose a Deep-learning based solution that can deal with forest fire disasters, that solution should evaluate and classify every image related to the forest environment to tell if that image is clean, and localize the place of fire in case there is fire or smoke.

Since our thesis aims to propose a deep learning solution, there are a lot of theoretical aspects that should be explained briefly, our thesis contains 4 chapters that include three main parts.

The first part is concerned with the theoretical part, starting from the Artificial intelligence passing by Machine learning, and ending with Deep Learning where:

- Chapter 1: represent a brief overview of the Artificial intelligence and Machine

learning field.

- Chapter 2: an overview of most used Deep learning techniques, where we have explained how CNN works and stated some of the most important CNN architectures.

The second part is concerned with the history of Object recognition and some of the previous research in that field concerning the fire detection problem.

In the third part (practical part) of this thesis, we will try to generate a custom images dataset from the environment of "Ain Mimoune Forests" using custom algorithms, then we have proposed a new deep-learning-based solution that implements an ensemble-learning strategy for forest fires detection, this solution into consideration the limitations of resources and real-time system performances, by integrating two of the most powerful CNN models, the first one VGG16 to avoid false positives, and YOLOV5 to localize objects.

After completing the proposed solution, we have tried to create a forest-simulation environment using the Unity3D game engine, which will help us test our algorithm on a forest-similar environment, this step has opened a big door for future perspectives and suggestions that can be done concerning the fire detection problem.

In the end, we made a web application interface to make our solution easily used by anyone using the Django Python framework, since the backend language of our application is python, it was easy to convert and use our proposed algorithm directly.

Chapter 1

Artificial Intelligence and Machine Learning

1.1 Introduction

The field of computer science has witnessed a great development in all aspects from software engineering to the hardware industry branches, which has led to huge improvements and new programming approaches that solve real-world problems.
One of the most important of these branches is the Artificial Intelligence branch which aims to create intelligent machines that can solve problems that could not be dealt with classic programming approaches.

1.2 Artificial intelligence

1.2.1 Definition

Artificial intelligence (AI) is the science that empowers machines to think, using technology and algorithms to help computers solve problems. Most AI products you know today from chat boxes to self-driving cars and manufacturing robots rely heavily on deep learning (neural networks) and machine learning. These techniques can be used to train computers to perform specific tasks by processing large amounts of data and identifying patterns in the data. [1]

1.2.2 A Brief History of Artificial intelligence

The history of artificial intelligence dates back to ancient times when philosophers pondered the idea that artificial beings, robotic humans, and other automata existed or might exist in some way.

AI became more tangible in the 17th century and beyond, thanks to early thinkers. Philosophers ponder how human minds can be mechanized and manipulated by intelligent non-human robotic workers. The thought processes that sparked interest in AI emerged when classical philosophers, mathematicians, and logicians considered the (mechanical) manipulation of symbols, culminating in the invention of the Atanasoff Berry Computer (ABC), a programmable digital computer, in the 1940s. This particular invention has inspired scientists to push the idea of creating ”electronic brains” or artificially intelligent creatures.

It was nearly a decade before the icons in AI helped make sense of the fields we have today. Mathematician Alan Turing proposed a test that measures the ability of machines to replicate human behavior to an indistinguishable degree from human behavior. Later in that decade, the field of AI research was established at a summer meeting at Dartmouth College in the mid-1950s, where computer and cognitive scientist John McCarthy coined the term ”artificial intelligence”.

Beginning in the 1950s, many scientists, programmers, logicians, and theorists helped solidify the modern understanding of artificial intelligence as a whole. Each new decade brings innovations and insights that change fundamental understandings of the field of AI

and how historical progress has propelled AI from an unattainable fantasy to a tangible reality for present and future generations[2].

1.2.3 Artificial intelligence applications

Artificial intelligence has been dominant in various fields such as:

1.2.3.1 Game artificial intelligence:

Playing a key role in strategy games, machines can find a large number of possible positions based on heuristic knowledge, e.g. poker, chess, etc.

1.2.3.2 Expert systems:

Some applications integrate machines and software that allow reasoning and offer advice to users.

1.2.3.3 Intelligent robots:

Robots are machines or devices that operate automatically or by remote control. They have sensors to detect the physical data of the real world, such as temperature, heat, light, movement, sound, shock, pressure, etc. They are able to learn from their mistakes and adapt to a new environment in order to accomplish the tasks entrusted to a human [3].

1.2.3.4 Voice recognition:

Some intelligent systems are able to hear and understand the human voice. These systems allow to analyze the language in terms of sentences and their meanings while a human is speaking it.

1.2.3.5 Vision systems:

these systems interpret and understand visual inputs to the computer. For example,
- Doctors use a clinical expert system to diagnose the patient.
- A spy plane takes photographs that are used to understand spatial information.
- The police use a computer software capable of arresting a person criminal through facial recognition systems.

1.3 Machine learning

1.3.1 Definition

Machine learning is a branch of computer science that is concerned with building algorithms in order to be useful, rely on collections of examples of phenomena. These examples can come from nature, handcrafted by humans, or generated by other algorithms. It can also be defined as the process of solving real-world problems by collecting

a dataset and algorithmically building statistical models based on that dataset. It is believed that this statistical model is somehow used to solve real problems.

1.3.2 Machine learning techniques

Techniques of machine learning are fundamentally algorithms and this can be work on data and retrieve insights from it, this can include discovering, predicting, or forecasting patterns and trends. The concept is to develop a model utilizing a data combination and algorithms that can be utilized to work on new before unseen derive and data actionable insights. Every technique relies on what type of data this can work on and the objective of issues they are trying to sort out. People generally get convinced to learn an algorithms couple and after that attempt to apply them to each issue. An essential point to remember is that there is no any universal machine learning algorithm that fixes complete issues.

The main inputs for the machine learning algorithm are aspects that are retrieved from the data utilizing a procedure known as the aspects extractions, this is generally coupled with the other process named aspect engineering or developing new features from the existing features. Every feature can be explained as an aspect of the dataset, like their locations, age, number of shares posts, and many more, if dealing with the data related to the social media profiles of the users. Techniques of machine learning can be categorized into four main types called supervised, unsupervised learning, reinforcement learning and semi-supervised learning [4].

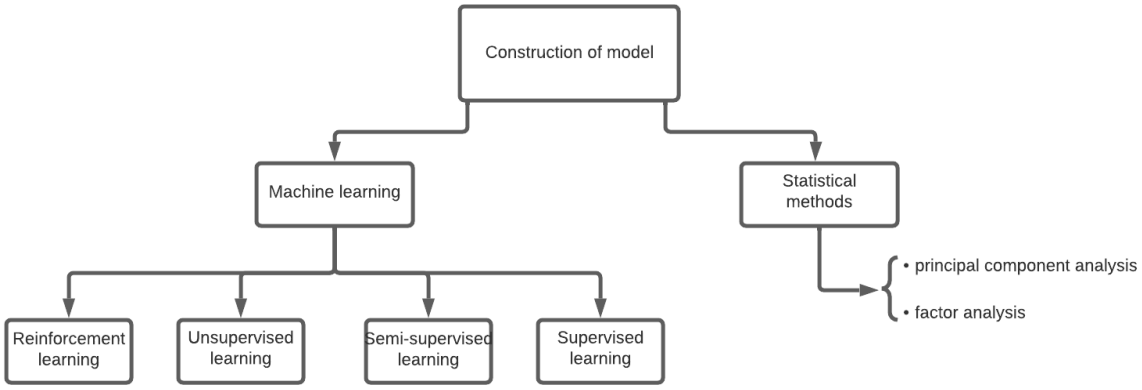

Figure 1.1: Different techniques from AI

1.3.2.1 Supervised learning

A type of machine learning is considered supervised when the dataset used in the process is already classified or contains examples with relevant correct classes and the algorithm needs to use it to classify the outcome to be able to do so later when the data is no longer classified. For instance, when learning to classify handwritten digits a supervised learning algorithm requires a lot of handwritten images and classes related to each image. The algorithm tries to find the patterns between images and their associated classes and applies the learned patterns to predict entirely new images (without labels) that the machine has never seen before [5].

To demonstrate how supervised algorithms works, let's analyze the problem of forecasting annual salary based on the number of accomplished educational years. formally, we'd like to create a model that approximates the association between the accomplished educational years X and the corresponding annual salary Y.

$$Y = f(x) + \epsilon$$

X (input) = *accomplished educational years*
Y (output) = *annual salaray*
f = *function describing the relationship between X and Y*
ϵ = *random error term (positive or negative) with mean zero*

In supervised learning, the machine tries to comprehend the association between salary and education years, by running classified training data via a learning algorithm. The resulted model can be used to predict the salary of people whose salary Y is unspecified, as long as we have accomplished educational years X as inputs. In different terms, we can use our model to predict the annual salary Y using unlabeled test data X (accomplished educational years) [6]. The workflow of supervised machine learning algorithms is given in figure below (figure 1.2):

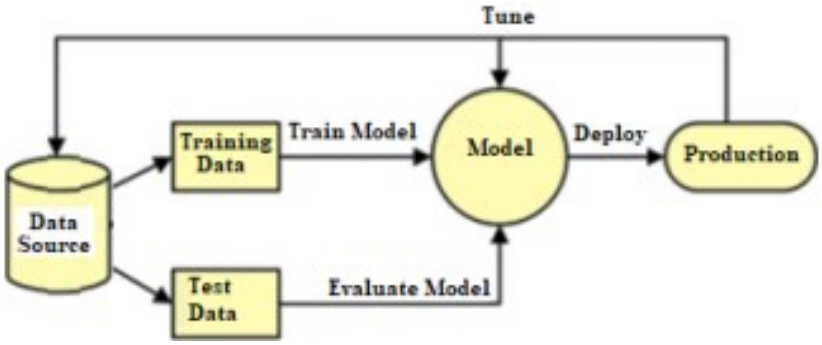

Figure 1.2: Supervised learning workflow [7]

Here are some algorithms that can be used as a supervised learning:
- Decision Tree
- linear regression
- logistic regression
- K-NN *(K nearest neighbours)*
- SVM *(Support vector machine)*

1.3.2.2 Unsupervised learning

Unsupervised learning is much more complicated because the system needs to identify similarities in the data it receives and organize them according to the similarities. This way of working has great advantages in the classification phase [5].

When using unsupervised learning, we don't care about the target output because the goal of the algorithm is to find relationships in the data and group data points based only on the input data. Supervised learning involves labeling data for prediction purposes, but unsupervised learning does not.

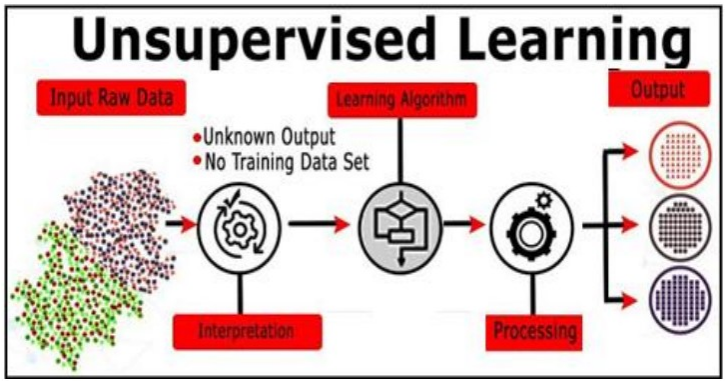

Figure 1.3: Unsupervised learning workflow [7]

Here are some of the most important unsupervised learning algorithms (Clustering) :

- K-Moyens.
- k-means.
- Hierarchical Cluster Analysis (HCA).
- Visualisation et réduction de la dimensionnalité.
- Principal component analysis (PCA).
- T-distributed Stochastic Neighbor Embedding (t-SNE).
- Apprentissage des règles d'association.

1.3.2.3 Renforcement learning

Reinforcement learning is a class of computational algorithms that specifies how an artificial agent (e.g., a real or simulated robot) can learn to select actions to maximize the total expected reward. This computed difference, termed reward-prediction error, has been shown to correlate very well with the phasic activity of dopamine-releasing neurons projecting from the substantia nigra in non-human primates.

Reinforcement learning is a special case of supervised learning, where the exact desired output is unknown. The teacher supplies only feedback about the success or failure of an answer. This is cognitively more plausible than supervised learning since a fully specified correct answer might not always be available to the learner or even the teacher. It is based only on the information as to whether or not the actual output is close to the estimate. Reinforcement learning is a learning procedure that rewards the neural network for its good output result and punishes it for the bad output result [8].

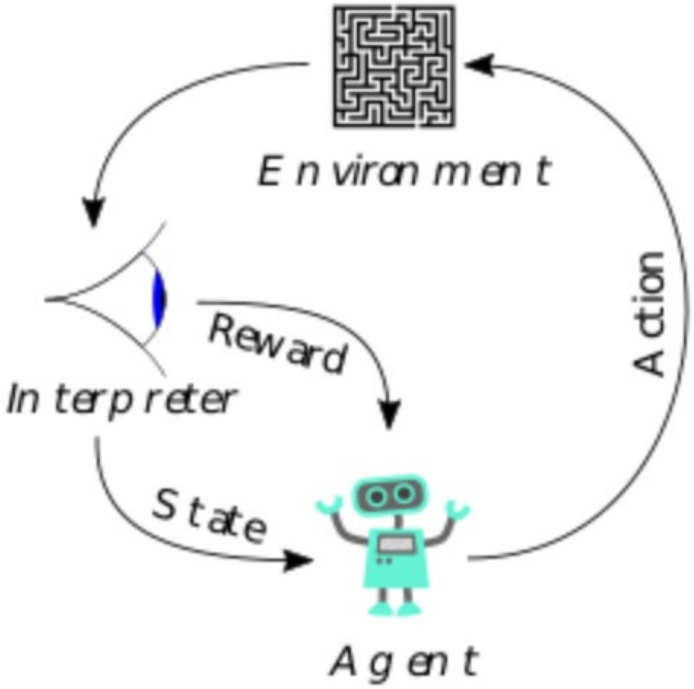

Figure 1.4: Renforcement learning workflow [7]

1.3.2.4 Semi-supervised learning

There are other types of classification based on other types of learning methods such as “semi-supervised learning” In effect, semi-supervised learning is a good compromise between the two types of “supervised” and “unsupervised” learning because it makes it possible to process a large amount of data without needing to label them all, and it takes advantage of the advantages of both types mentioned.

On the other hand, a priory labeling of all the data requires the intervention of a human expert. This is a difficult or even tedious operation when the number of data is important. In concrete applications, it is often impossible for the expert to assign all the training data to the classes present [9].

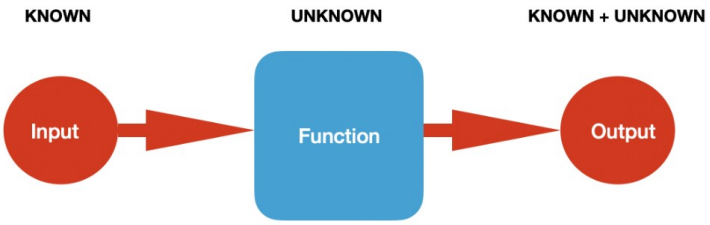

Figure 1.5: Semi-supervised learning

1.3.3 Tasks of machine learning

The tasks of machine learning algorithms and when you should use each of them. I particularly think that getting to know the tasks of Machine learning algorithms is like

getting to see the Big Picture of AI and what is the goal of all the things that are being done in the field and put you in a better position to break down a real problem and design a machine learning system.

1.3.3.1 Regression

Regression is used to estimate the value of a label from a collection of related features. Labels can be any ground-truth value, rather than from a limited set of values as in classification tasks.

Regression algorithms relate the dependencies of a label to its associated features to indicate how the label changes when the feature value changes. A regression algorithm takes a set of samples with known label values and provides a model that predicts the label values for a new given set of features [10]. Here are some examples where regression can be used:

- Estimate home prices based on a range of characteristics such as number of rooms, community composition, or home size and usable area.
- Predict Bitcoin price based on historical data and current market trends.
- Forecast product sales based on advertising budget [10].

1.3.3.2 Classification

For the classification tasks, the algorithm takes a set of labeled samples and outputs a classifier that we can use to predict the class of new unlabeled instances, where is the number of classes is limited. Classification can be used for:

- Classify face expressions as 'happy', 'sad' or 'angry'.
- Classify bitcoin as "Save" or "Danger" to buy using current market trends.

1.3.4 Approaches of machine learning

Here is a some and famous machine learning approaches bellow that We can decide which one should select based on the problem statement.

1.3.4.1 Decision tree learning

That type of learning uses decision tree algorithms to connect the input features with the target class.

1.3.4.2 Association rule learning

Learning association rules is a way to extract important associations between variables of big databases.

1.3.4.3 Learning Algorithms

Learning algorithms are the most popular and fundamental algorithms in artificial intelligence. these types of algorithm can adapt to problem situations [11].

1.3.4.4 Artificial neural networks

Artificial neural network (ANN) is also known as a connectist system. ANN is inspired by the brain's neural network of the living organism. ANN is a framework of various machine learning algorithms which work together to process complex dataset. As shown in figure 1.6 every ANN can be divided into three layers: the input layer, hidden layers, and the output layer. Hidden layers can have multiple layers. Every layer is a collection of nodes which are called artificial neurons. Each artificial neuron is connected to the artificial neurons present in the next layer. The connection between these neurons is called edges. These edges transmit the information as a signal from one neuron to another, a neuron then processes the information and carry forward it to the next neurons [11].

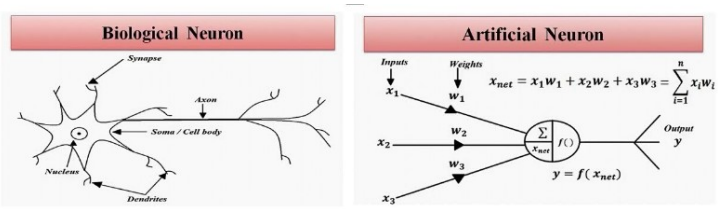

Figure 1.6: Artificial neuron biological neuron[12]

1.3.5 Conclusion

AI is still a young field, but it is still very vast, It has been subject to several criticisms since the beginning of its creation but has still been able to develop to the point of achieving quite surprising results.
AI has even more room to develop further. Indeed it shows a great potential to facilitate the life of every day and even explore horizons that were not reachable before.
This chapter has just been a brief overview of AI and at the same time an introduction to the second chapter of our theme entitled deep learning.

Chapter 2

Deep learning

2.1 Introduction

Deep Learning has been a challenge to define for many experts in the field as it has slowly changed shape over the past decade. One useful definition states that deep learning is a neural network with more than two layers [13]. The problem with this definition is that it echoes the existence of this field since the 1980s of the last century, forming a large contradiction since many people think this field is relatively new, to refute this contradiction, one must refute this contradiction, we need to distinguish between when the field emerged and when it was framed and exploited.

2.2 Definition

Deep learning is a branch of machine learning that deals with algorithms inspired by the networks and operations of the brain, named artificial neural networks. By way of explanation, it mirrors how our brains work. Deep learning algorithms are closer to the structure of a nervous system, where each neuron is connected and transmits information. Deep learning models work in layers, and a standard model has at least three layers. Every layer takes information from the previous layer and passes it to the next layer [14].

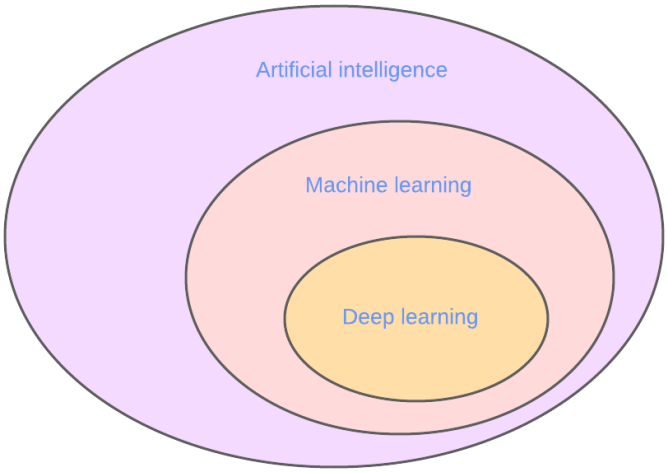

Figure 2.1: The sub-branches of artificial intelligence

2.3 Difference Between Machine learning and Deep learning

Deep learning models usually function well on datasets, while older machine learning models stop enhancing after reaching a saturation point.

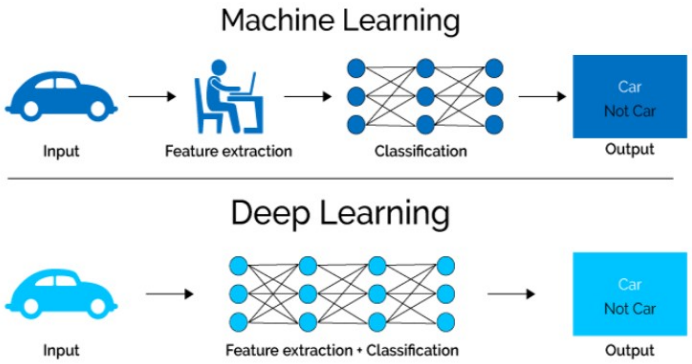

Figure 2.2: Feature extraction area in machine deep learning [15]

One of the differences between machine learning and deep learning models is in the field of feature extraction. Feature extraction is done by humans in machine learning, while deep learning models compute it themselves.

2.4 Application areas of deep learning

These techniques are developed in the field of computer science applied to NTIC (visual recognition - for example of a road sign by a robot or an autonomous car and voice recognition), robotics, bioinformatics, pattern recognition or comparison of forms, security, health, etc, computer-assisted education, and more generally to artificial intelligence.

Deep learning can for example allow a computer to better recognize highly deformable objects and/or analyze for example the emotions revealed by a photographed or filmed face, or analyze the movements and position of the fingers of a hand, which can be useful to translate sign language, to improve the automatic positioning of a camera,etc.

They are used for some forms of medical diagnostic assistance (e.g., automatic recognition of cancer in medical imaging), or prediction (e.g.: prediction of the properties of a soil filmed by a robot).

2.5 Deep learning architecture

Deep learning is a neural network with a large number of parameters and layers, the basic example is the multilayer perceptron.

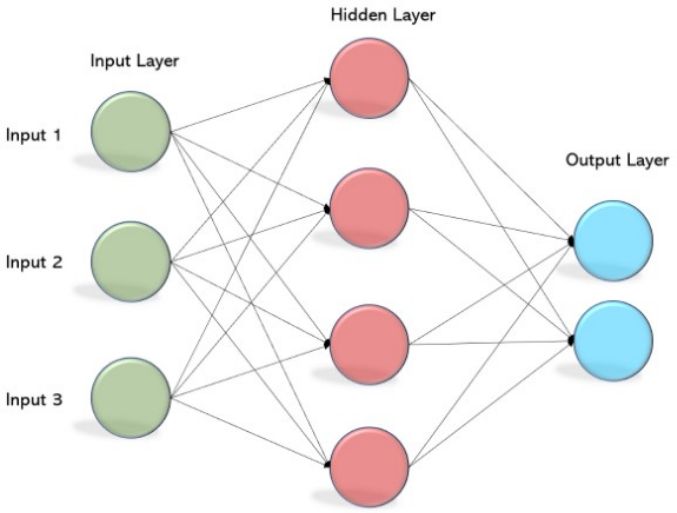

Figure 2.3: Multi-layer perceptron [16]

There are a large number of deep architecture variables. Most of them are derived from some original architectures. We will start with the convolutional neural networks (CNNs).

2.5.1 Convolutional neural networks

Convolution neural network was introduced by Yann Lecun in November 1998 [17]. CNNs are a specific type of deep neural network model that is mainly used to recognize images because they are good at extracting features from Images that are invariant to rotation and change in scale (to an extent only through, you still require a very variant and large dataset to tackle translational variance), As CNNs are the heart of all the other architectures we are going to explain it deeply.

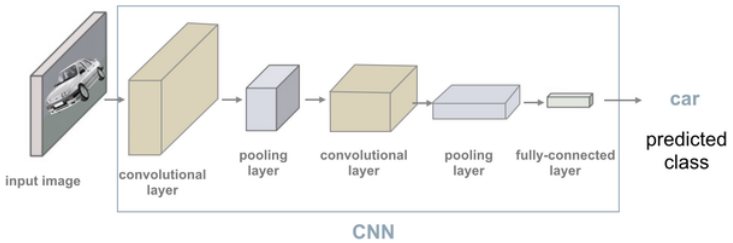

Figure 2.4: CNN architecture [18]

2.5.1.1 Convolutional layer:

From its name, we deduce the first word ”convolution”, which means that the network uses a mathematical operation called convolution. This is a special type of linear operation. The main goal of convolution is to extract features such as edges, colors, and vertices from the input. As we dig deeper into the network, the network also begins to recognize more complex features, such as shapes, numbers, and parts of faces. But how does this layer extract features such as edges, colors, etc. To understand how convolutional layers extract these features, let’s first understand how our machine sees images.

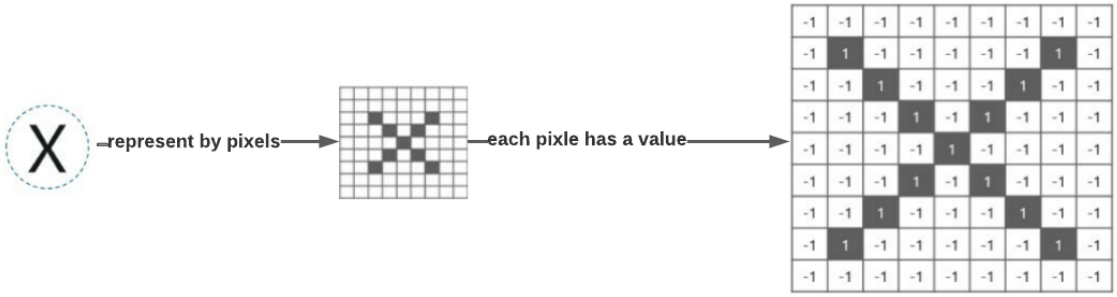

Figure 2.5: How machines understand images

Every image or picture in our machines represent by pixels end every pixel has a value of the color scale between 0 and 255, 0 means this pixel is black and 255 means this pixel is white, and so on.

In grayscale images, each pixel represents the intensity of only one shade how bright or dark the pixel is. In other words, it has only one channel. On the other hand, for colored images, we have three color channels: R, G, and B (red, green, blue). Standard digital cameras have three (RGB) channels.

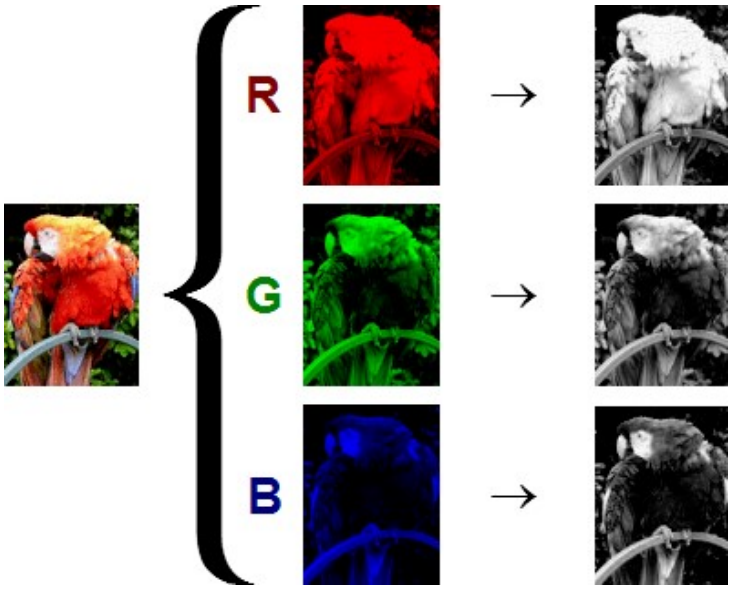

Figure 2.6: How do machines understand colored images[19]

As shown in the image above (figure 2.6), a colored image consists of three channels: red, green, and blue. The question currently is, how does the computer see that picture? The answer is that they saw the matrix.

How can we represent this picture in a matrix since it has three channels instead of a grayscale image where we only have one? In this case, we have a three-dimensional matrix. Each channel has a two-dimensional matrix, and the three matrices for the red,

green, and blue channels are stacked on top of each other, making the matrix a three-dimensional shape.

For example, the shape of a matrix representing a 500px by 500px color image will be (500, 500, 3). Each pixel in this color image has three numbers (ranging from 0 to 255) associated with it. These numbers represent the intensity of red, green, and blue colors in that particular pixel.

After understanding how exactly picture representation is, let's dive into how the convolutional layer extracts the features from images. A convolution is a transformation pixel by pixel, done by applying to an image some transformation defined by a set of weights, also known as a filter or kernel. Let's be a set of source pixels, and w a set of weights. A pixel y is transformed as follows:

$$\sum_{i=0}^{n} Si.Wi$$

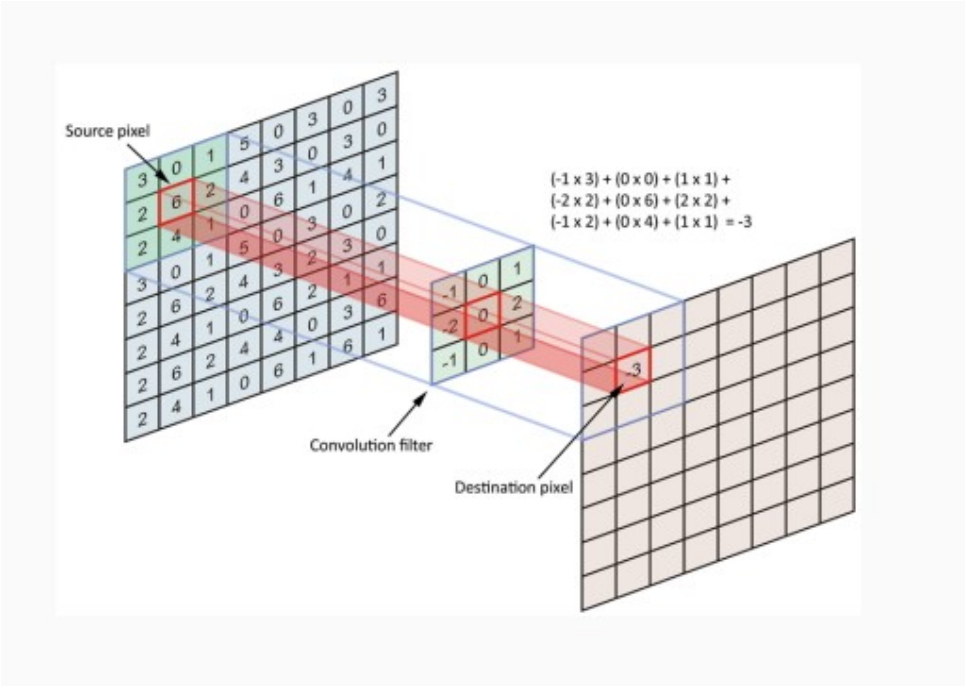

Figure 2.7: Convolution operation [20]

We are talking about things like straight lines, plain colors, and curves when we say features. Consider the most basic traits that all images share. These features are recovered by shifting the kernel over the image and applying a matrix multiplication between the kernel and the area of the image where the kernel is hovering at any given time.

The filter moves to the right by a determined step value or step size (**Stride**) until the entire width is resolved. It continues to the beginning of the frame (left) with the same step value and repeats the process until the entire frame is traversed.

2.5.1.2 Polling layer:

Pooling layers, like convolutional layers, are responsible for minimizing and reducing the size of convolutional features. The computational power required to process the data is reduced through dimensionality reduction. It also facilitates the extraction of rotation and position invariant main features, which allows the model's training process to run smoothly. Pooling can be divided into two types: maximum pooling and meduim or average pooling. Maximum pooling returns the maximum feature or value of the piece of the picture surrounded by the kernel. On the other hand, average pooling returns the average of all values from the kernel piece of the picture. Max pooling can also act as a noise canceller. It removes all sound activation and performs well.

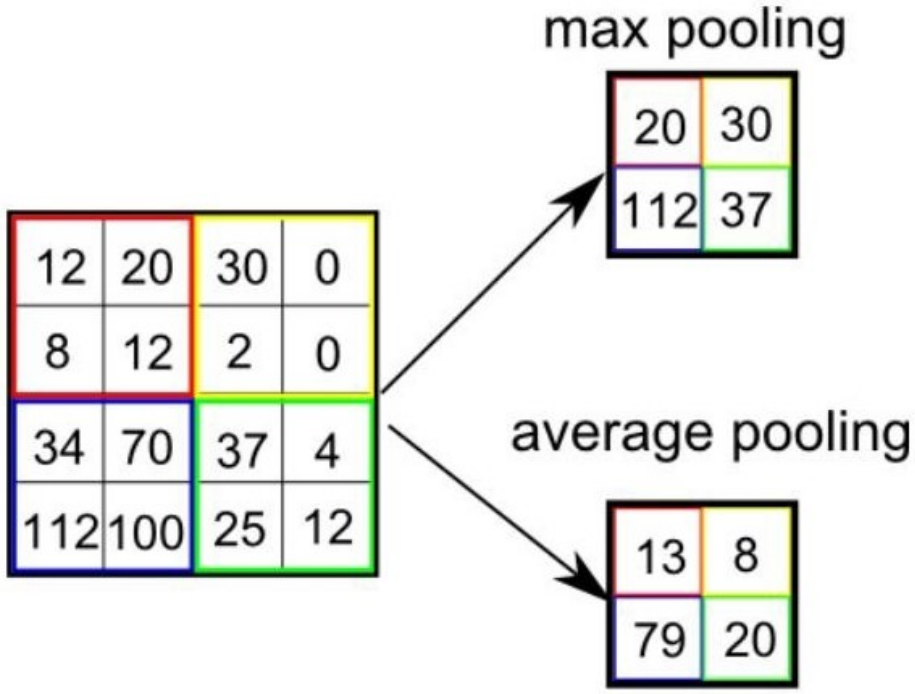

Figure 2.8: Types of polling operation [21]

2.5.1.3 Fully connected layer (dance layer):

We haven't done anything yet to categorize the various images. We have just highlighted specific characteristics in an image while substantially reducing its size. These high-level features can now be detected by Connecting Fully connected layers (dense layers) to the network's end. This layer takes an input (the result of the conovolution, ReLU, or pool layer) and returns an N-dimensional vector, where N represents the class the numbers from which the program must choose.

For example, if we are dealing with the digits recognition problem, N would be ten because there are ten digits. The likelihood of each class is represented by a number in this N-dimensional vector. For example, if the output is [0, 0.1, 0.1,0.75, 0, 0, 0, 0, 0,

0, 0.5,], it means that there is a 10% chance that the number is a 1, a 10% chance that the image is a 2, a 75% chance that the image is a 3, and a 5% chance that the image is a 9. (As a side note, there are numerous ways to describe the output; I'm simply demonstrating the softmax method).

This fully linked layer analyzes the output of the preceding layer (which should represent the activation maps of high-level qualities) and determines which features are most associated with a specific class. For example, if the computer is predicting that a certain image is a mouse, the activation maps will contain high weights or values that reflect high-level traits such as a foot or four legs, and so on.

Also, if the algorithm predicts that a given image is a bird, the activation maps that reflect high-level traits such as wings or a beak will have high values. A fully connected layer considers which high-level traits are most strongly associated with a specific class and assigns weights to them, resulting in the right probabilities for the various classes when the products of the weights and the preceding layer are computed [22].

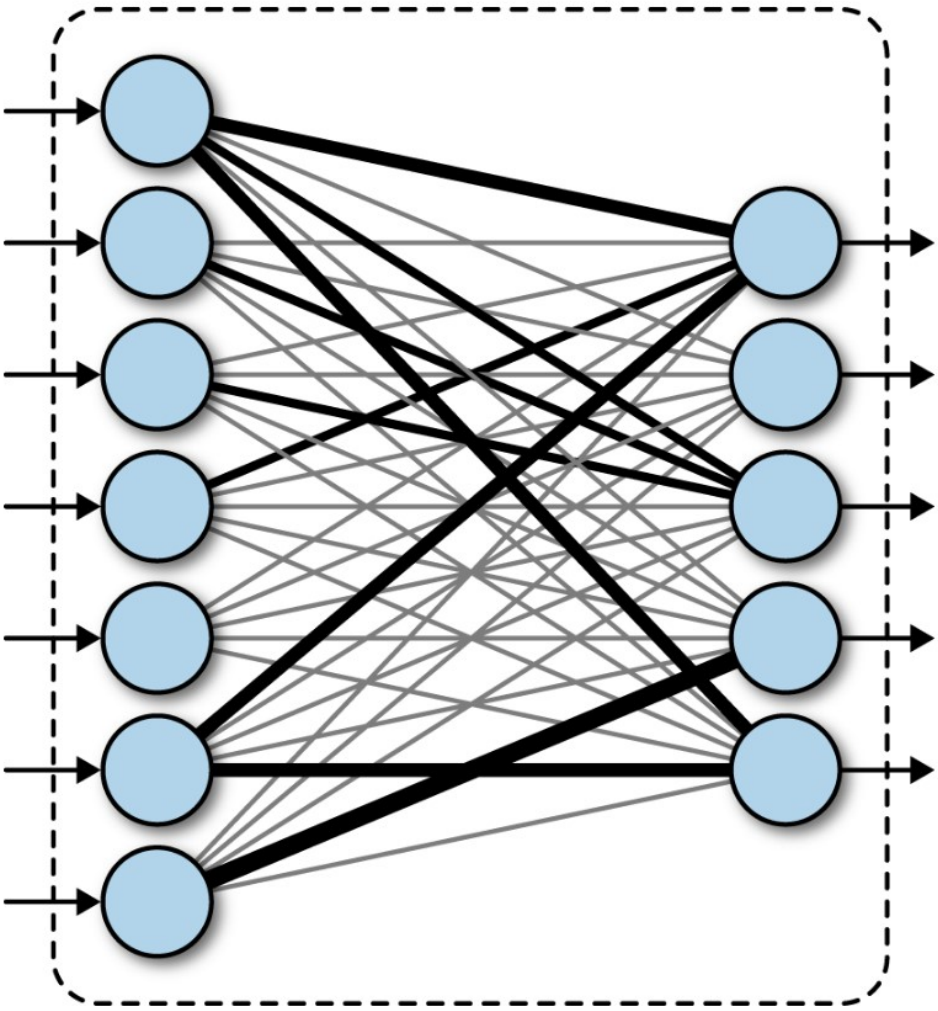

Figure 2.9: Fully connected layer [23]

2.5.2 R-CNN

R-CNN (Region-based Convolutional Neural Network) is a deep learning model for computer vision and image processing. The basic purpose of any R-CNN built for object detection is to find objects and define a boundary around them in any input image. As we will see in the picture bellow (figure 2.10):

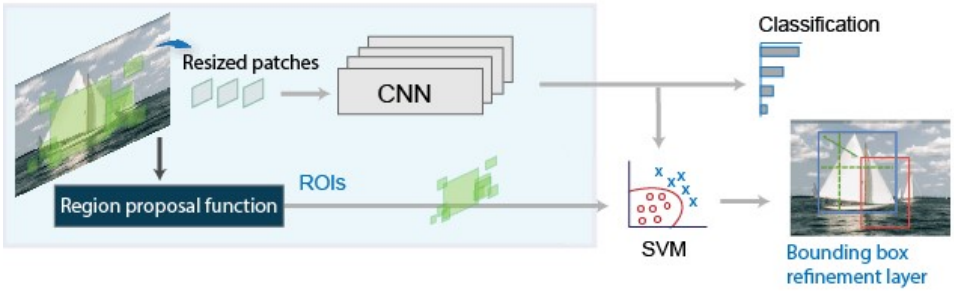

Figure 2.10: R-cnn [24]

R-CNN was proposed by Ross Girshick and the algorithm can be divided into three steps [25]:

1. Candidate region extraction. R-CNN adopts a higher quality region extraction method, namely region proposal based on selective search (SS). Generally, 1k 2k candidate frames are extracted from the image to be detected.

2. CNN feature extraction. After the candidate frame is scaled to a fixed size, the feature of each candidate frame is extracted by CNN to obtain a feature vector of a fixed dimension.

3. Classification and boundary regression. The obtained eigenvectors are input to SVM for classification, and the category information is obtained and then sent to the fully connected network for boundary regression.

Although R-CNN is cleverly designed to break the bottleneck of the DPM algorithm for many years, it seems that it has many shortcomings: First, although the "exhaustive method" of the sliding window is avoided, R-CNN still has repeated operations and thousands of candidates. The box needs to extract features through CNN, the amount of calculation is still large, and the overlap of candidate frames brings a lot of unnecessary double counting.
Secondly, the training process of R-CNN is very complicated. Candidate region extraction, feature extraction, classification, and regression are operated separately, and the data between steps are saved separately, which results in its slow speed. Finally, the scaling of input before feature extraction of CNN will cause image distortion, and the detection performance of the algorithm will be greatly reduced [26].

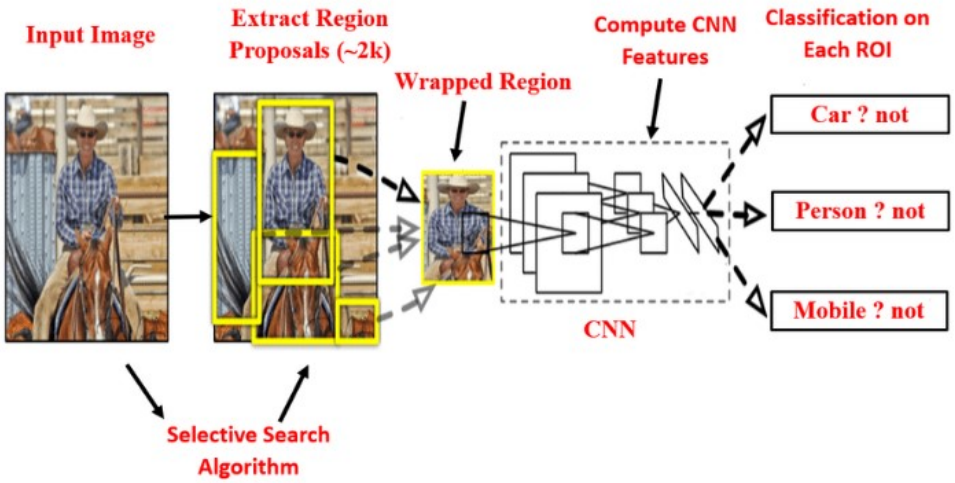

Figure 2.11: R-CNN architecture [25]

Our system takes an input image, extracts about 2000 region proposals, computes features for every region using a large convolutional neural network, and then classifies each region using a class-specific linear SVM. R-CNN achieves 53.7% mean precision (mAP) on PASCAL VOC 2010.

The comparison shows that 35.1% of mAPs use the same region proposals, but with the spatial pyramid and high visual word methods. The popular model with ductile parts achieved 33.4% performance. On the ILSVRC2013 200-class detection dataset, R-CNN achieves an mAP of 31.4%, a large improvement compared to the OverFeat's best result of 24.3% [25]. However, R-CNN is so sluggish because it runs each object proposal through a ConvNet forward pass without mentioning the high number of 2000 region proposals [27].

2.5.3 Fast R-CNN

The entire image and a set of object proposals are sent to the Fast R-CNN network. To generate convolutional feature maps, the network first analyzes the whole image using convolutional and max-pooling layers. The Region of Interest (RoI) layer then derives a feature vector with a defined length from the feature map of each object proposal. Each feature vector is fed to a series of fully connected (FC) layers that split into two output layers with siblings: one that generates softmax probability estimates for the K object classes, and the other is the "background" class. For each of the K object types, there is a layer that returns four real numbers. The improved bounding box position for one of the K classes is encoded by each set of four values [27].

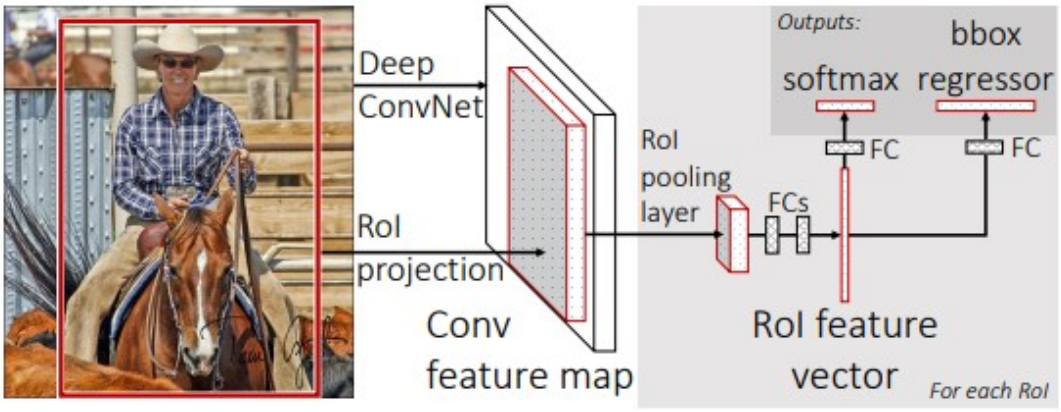

Figure 2.12: Fast R-CNN architecture [27]

A fully convolutional network is fed the input image and several regions of interest (ROIs). Each ROI is combined into a single feature map with a set size, which is then mapped to a feature vector via a fully connected layer (FC). For each class of the bounding box, the network produces two output vectors: the softmax probability and regression offset. A multi-task loss is used to train the architecture from beginning to end [27].

2.5.4 Faster R-CNN

Faster R-CNN is made up of two modules. A deep fully convolutional network proposes regions in the first module, while a fast R-CNN detector uses proposed regions in the second. The system as a whole is one unified object detection network (Figure 2.13). The RPN module informs the Fast R-CNN module where to look, using the increasingly popular name for neural networks with an ”attention” mechanism [28].

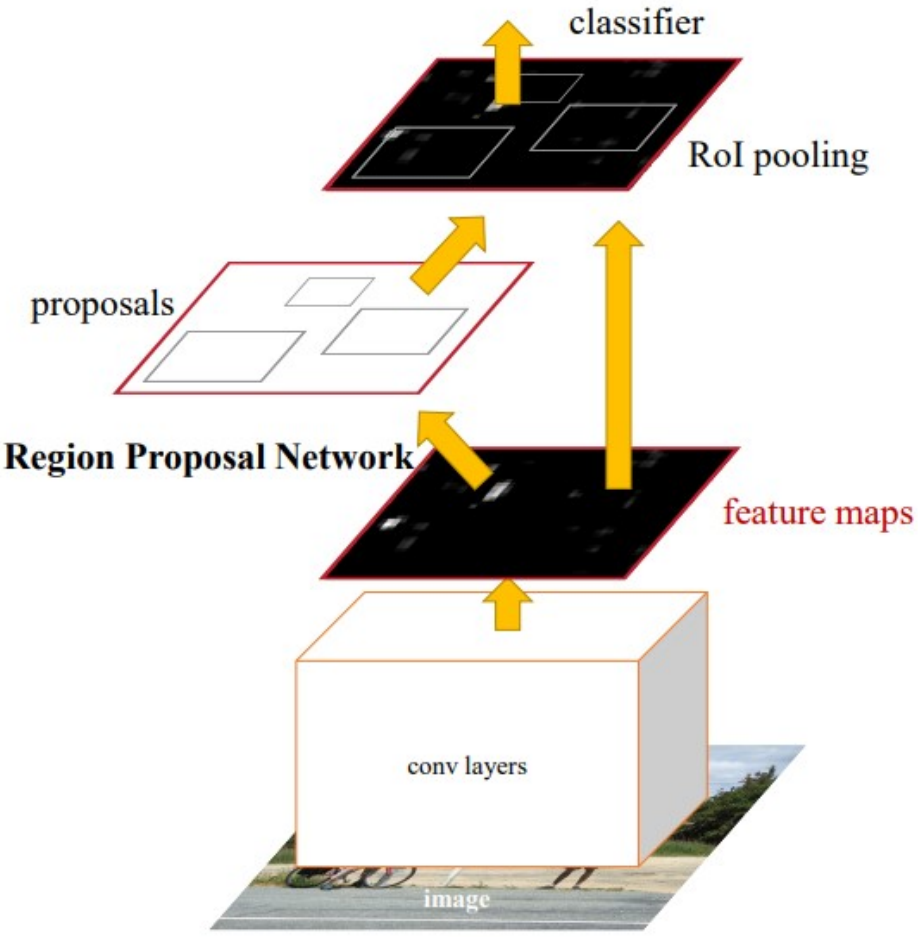

Figure 2.13: Faster R-CNN architecture [28]

1. **Conv layers:** In these layers, we train filters to extract the corresponding features of the image. For example, let's say we want to train these filters to extract features suitable for faces. Then simply by training, these filters will learn the shapes and colors present in the face.
2. **Region Proposal Network (RPN):** An RPN is a small neural network that slides over the last feature map of a convolutional layer and predicts the presence or absence of objects and predicts bounding boxes for those objects.
3. **Classes and Bounding Boxes prediction:** Now we use another fully connected neural network that takes as an input the regions proposed by the RPN and predicts object class (classification) and bounding boxes (regression).

2.5.5 YOLO

YOLO is a novel method to detect the presence of multiple objects in pictures in real-time and draw bounding boxes around them simultaneously. It just runs the image once through the CNN algorithm to get the output, which is the name. Although R-CNN is similar, YOLO runs much faster than Faster R-CNN due to its simpler architecture [29].

2.5.5.1 YOLOV5 architecture

There is no paper on YOLOv5 as of Jun 06, 2022, based on the yolov5 github repository: `https://github.com/ultralytics/yolov5/issues/1333`. Therefore, in this thesis we are going to elaborate on YOLOv4 so that it's easy to understand YOLOv5. To get a deep understanding how exacly Yolov5 improved the performance and its architecture, let us go through the high-level Object detection architecture presented in the figure 2.14 bellow:
YOLOv5 is, like many neural networks dedicated to image processing, based on layers of convolutions. The architecture of YOLO5 can be separated into three parts, as shown in figure 2.14.

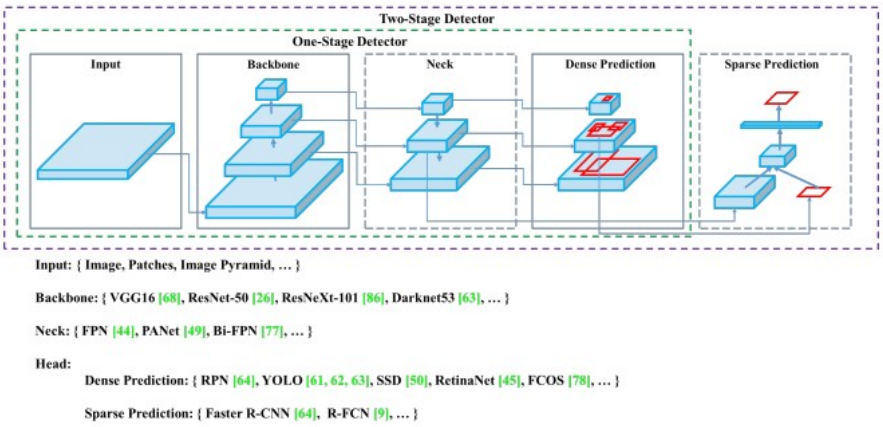

Figure 2.14: The network architecture of Yolov5 [30]

These three parts can be connected to a set of layers, making up a functional unit. To each of these parts therefore corresponds a role, established by this set of layers.

- **Input:** This box in Figure 10 does not really correspond to a part of the network as presented just above. It simply corresponds to the data taken as an input by the network, namely an image, a batch of images, even a video.

- **Backbone:** The first part of the network is called the backbone. It simply corresponds to a CSPDarknet53 which is derived from a CSPNet (Cross Stage Partial Network), first described in [31]. It is a convolutionel neural netrwork which it's main role is feature extraction.

- **Neck:** The second part of the network is called the Neck. It is in fact the assembly of two sub-parts: an SPPNet (described in [32]) and a PANet (described in [33]). The SSPNet (for Spatial Pyramid Pooling) makes it possible to get rid of fixed size arbitrarily in the architecture of a convolutional network, by a set of 12 consecutive poolings. In addition to removing this requirement from networks, SSPNet also makes detection robust to object deformations, as well as partially obscured or cropped objects. Regarding the PANet (for Path Aggregation Network), it corresponds to an improvement for YOLO, which makes it possible to obtain better predictions. This is achieved through work on how information propagates within the network.

- **Dense prediction (head):** This last sub-part, called Dense prediction or head in the article [30], simply corresponds to YOLOv3 (described in [34]). This is clearly the heart and the working base on which YOLOv4 is based. YOLOv3 is a fully-fledged, self-contained network that achieves decent performance (at the time of release) with very short inference time.

Yolov5 almost resembles Yolov4 with some of the following differences:

- Yolov4 is released in the Darknet framework, which is written in C. Yolov5 is based on the PyTorch framework.

- Yolov4 uses .cfg for configuration whereas Yolov5 uses .yaml file for configuration.

2.6 Conclusion

Deep learning is probably the most interesting and effective field in terms of results. It opens a multitude of doors that previously were just a distant dream.
In this chapter, we have dealt with the most crucial points to our work that this technology has offered us our hands. there are still a considerable number of points that we have intentionally left to be efficient in this branch of AI.
For the next chapter we are going to talk about the state of art of forest fire and smoke detection, What did researchers in this field do before us? and how they do it. we are going to explain the previous works in this topic.

Chapter 3

State of art

3.1 Introduction

After talking about the most important parts of deep CNN networks and different known architectures. In this chapter, we will talk about the history of Object recognition briefly and some of the previous research in that field concerning the fire detection problem.

Object recognition is a fundamental computer vision topic that entails finding instances of visual objects of a specific class (such as humans, animals, or cars) in digital images. Object recognition's purpose is to create computer models and approaches that supply some of the most fundamental information needed for computer vision applications.

3.2 Object detection history

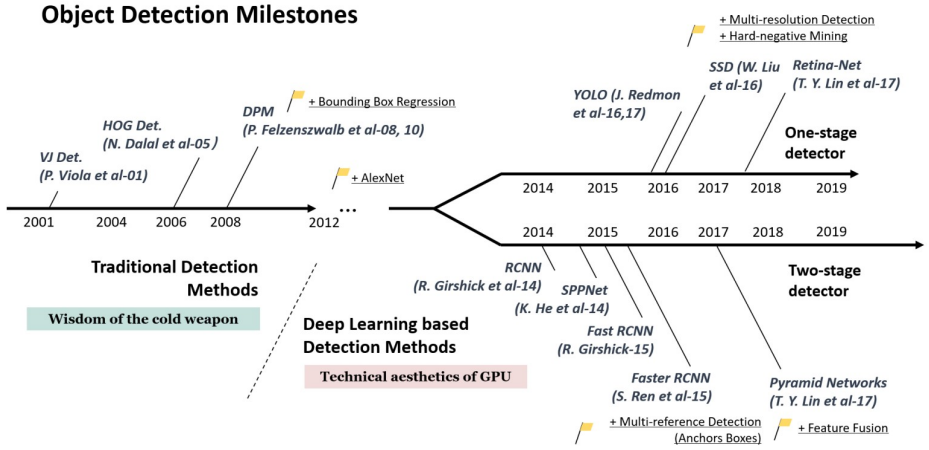

Figure 3.1: Object detection. Milestone [35]

The evolution of object detection has usually gone through two historical stages in the last two decades: the "classical object detection period (before 2014)" and the "deep learning-based detection period (after 2014)".

If we consider today's object detection to be technological aesthetics powered by deep learning, we'll be transported 20 years back in time to witness the "knowledge of the period of cold weapons." Most early object detection algorithms relied on constructed features due to a lack of good image representations at the time [35].

Viola Jones Detectors: Developed by Paul Viola and Michael Jones in 2001, this object detection framework enables real-time face detection. It uses a sliding window to iterate through all possible positions and scales in the image to see if the window contains a face. Sliding windows are essentially looking for "haar-like" features (named after Alfred Hall, who developed the concept of haar waves) [35].

HOG Detector : Hog was originally proposed by N. Dalal and B. Triggs in 2005 as an improvement to the scale-invariant feature transformation and shape context. HOG

works with a technique called block (similar to a sliding window), which consists of a dense grid of pixels where gradients are produced by the magnitude and direction of changes in the intensity of the pixels within the block. The applicability of HOG in pedestrian detection is well-known. To detect a wide range of objects of various sizes, it rescales the input image several times while keeping the detection window size the same [35].

Deformable Part-based Model (DPM): P. Felzenszwalb first introduced the DPM in 2008 as an expansion of the HOG detector. R has made a number of enhancements since then. The difficulty of detecting a "vehicle" window, body, and wheels can be broken down into a "divide and conquer" method. DPM uses this strategy. The training process involves learning a proper way of decomposing an object, and inference involves ensembling detections of different object parts [35].

Deep learning era: Unfortunately, after 2010, when the performance of handcrafted functions got saturated, object detection reached a plateau. Convolutional neural networks, or deep convolutional networks, were reborn in 2012, with deep convolutional networks learning resilient and high-level feature representations for pictures. Object detection was proposed by Region with CNN Capability (RCNN) in 2014 for object detection. Object detection is classified into two categories in the current era of deep learning: "two-step detection" and "one-step detection" [35].

3.3 Forest fire detection:

With rapid economic development, the increasing scale and complexity of constructions have introduced great challenges in fire control. Therefore, early fire detection and alarm with high sensitivity and accuracy are essential to reduce fire losses. However, traditional fire detection technologies, like smoke and heat detectors, are not suitable for large spaces, complex buildings, or spaces with many disturbances. Due to the limitations of the above detection technologies, missed detections, false alarms, detection delays, and other problems often occur, making it even more difficult to achieve early fire warnings.

Recently, image fire detection has become a hot topic of research. The technique has many advantages such as early fire detection, high accuracy, flexible system installation, and the capability to effectively detect fires in large spaces and complex building structures. It processes image data from a camera by algorithms to determine the presence of a fire or fire risk in images. Therefore, the detection algorithm is the core of this technology, directly determining the performance of the image fire detector. There are three main stages in the process of image fire detection algorithms, including image preprocessing, feature extraction, and fire detection. Among, feature extraction is the core part of algorithms. The traditional algorithm depends on the manual selection of fire features and machine learning classification. The shortcoming of algorithms is that manual feature selection should depend on professional knowledge. Though the researchers develop many studies on image features of smoke and flame, only simple image features, such as color, edges, and simple textures, are discovered. However, because of complex

fire types and scenes as well as many interference events in practical application, the algorithms extracting low and middle complex image features are difficult to distinguish fire and fire-like, thereby causing lower accuracy and weak generalization ability [36].

Because of this, to address this problem and identify fire and smoke, numerous research articles and methodologies have been offered. Some of the previously proposed methods are image processing algorithms for fire and smoke detection, motion-based smoke estimation, etc. Recently, many deep learning approaches have been proposed. It can detect fire and smoke automatically in videos and images.

3.4 Previous research

Starting with the paper [37], a new deep convolutional neural network algorithm is proposed for high-precision fire and smoke images. Rather than using tangent functions or standard rectified linear units, in the network's hidden layers, it utilizes adaptive piecewise linear units. A small database of fire and smoke images was also created to train and evaluate the model. To avoid overfitting problems caused by training networks with limited datasets, traditional data augmentation techniques and generative adversarial networks are used to increase the number of available training images. Experimental results show that the method produces high accuracy and a high detection rate, as well as a very low false-positive rate.

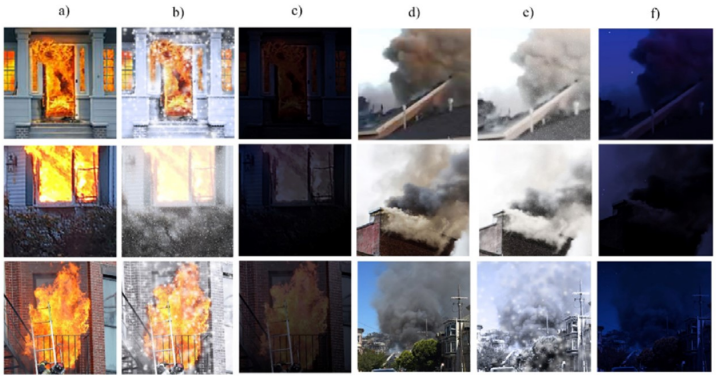

Figure 3.2: GANs generated samples where (a and d: originals, b, and e: generated) [37]

Deep CNNs	**Input image size**	**Original data**			**Augmented data**		
		AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)
AlexNet	227	93.41	92.85	0.86	96.15	95.25	0.57
ResNet	32	93.91	**93.15**	0.72	96.82	95.81	0.35
DenseNet	224	**94.66**	92.31	0.66	97.05	**96.57**	0.49
Article method	128	94.85	93.12	**0.63**	**97.15**	96.18	**0.33**

Table 3.1: COMPARISONS WITH OTHER DEEP CNNS [37]

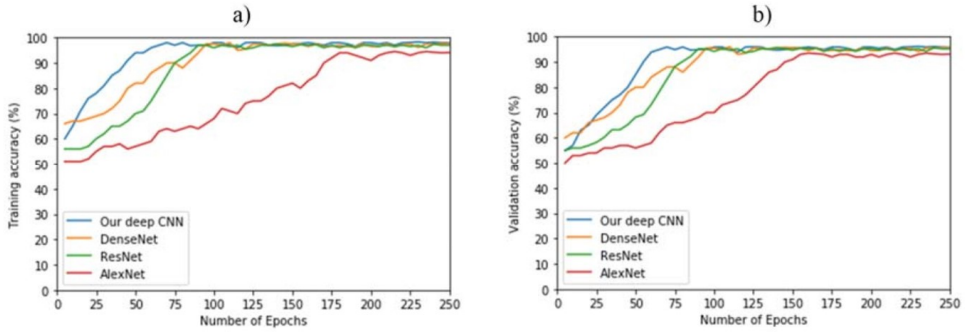

Figure 3.3: Training and validation curves of our deep CNN, DenseNet, ResNet, and AlexNet. a) The training curves, b) The validation curves. [37]

This research demonstrates that deep CNNs can achieve very high classification performance, even when limited data is provided. Overfitting problem caused by limited training image datasets leads to low-grade performance in neural network models. To solve this problem, we enlarge our training sets using various data augmentation techniques.
The use of GANs to create additional training samples improves the diversity of data available and helps the network to learn from fire and smoke features under different weather and light (day-time and night-time) conditions. Furthermore, The usage of GANs to make additional training simples enhances the diversity of data available and allows the network to learn from fire and smoke features under different weather and light (day-time and night-time) conditions. Furthermore, The issue right now is that the algorithm is unable to detect the exact location of fire and smoke.

Another paper under the name of DeepFireNet [38], which is a real-time fire detection system combining fire characteristics and convolutional neural networks, is proposed in this paper and can be used to detect video captured in real-time by monitoring equipment. DeepFireNet takes video streams from surveillance devices as input. To begin, a significant number of images that do not contain a fire in the video stream are filtered based on the features of fire. Extract suspected fire areas in images from fire images in video streams in the process and eliminate the influence of interference sources such as light sources and candles, and decrease the impact of complex environments on fire detection. Then, the algorithm encodes the extracted region, feeds it into the DeepFireNet neural network, extracts the image's depth information, and finally decides whether the image contains fire. The DeepFireNet network replaces the first layers that contain a 5x5 convolution kernel with two 3x3 convolution kernels, and the network's core architecture consists of only three improved initial layers, effectively reducing network parameters and greatly reducing computing cost. Experimental results show that the method can be applied to a variety of different indoor and outdoor scenarios. Besides, in the real-time video detection technique, the algorithm effectively meets the detection algorithm's precision and real-time performance requirements. This strategy is really practical.

Algorithm	ACC/%
VGG16	90.28
ResNet	91,8
InceptionV1	93.58
Inception V1-OnFire	93.85
DeepFireNet	96.86

Table 3.2: Accuracy of three algorithms [38]

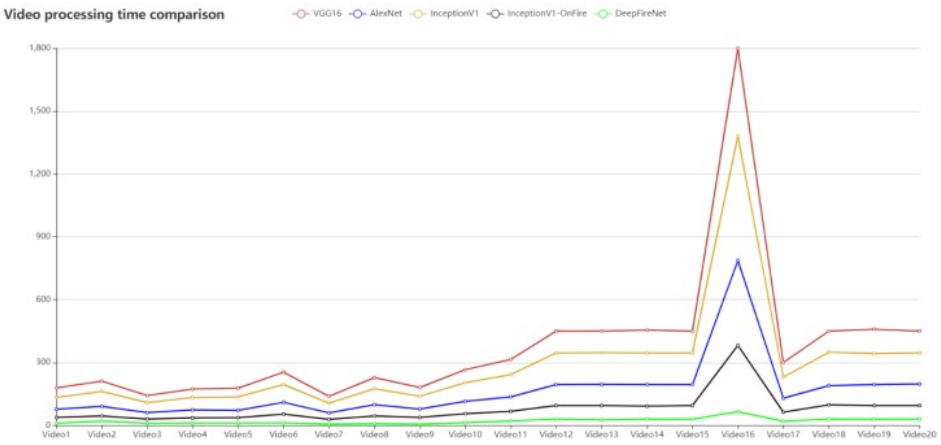

Figure 3.4: Comparison of processing time of algorithms on test sets. [38]

A fire detection algorithm with both detection accuracy and low time complexity is proposed. The algorithm has certain versatility and a high detection rate for fires in different stages. According to the real-time requirement in the video stream processing process and the interference of other complex environments such as light sources and fast-moving objects, In this work, a fire feature detection algorithm with extremely low time complexity is used to extract supposed fire areas, and a large number of non-fire images are filtered, and then, the detected fire suspicious area is input into the convolutional network to complete the fire identification of the area.
As the comparison table 3.2 shows, the algorithm greatly reduces the time complexity by filtering a large number of non-fire images and improving the inception layer convolution network, and greatly decreases the interference of difficult environments on the fire identification process by extracting the fire areas in the images so that the convolutional network only needs to focus on identifying fire features, effectively improving the recognition accuracy. Since fires often produce large amounts of smoke, in the following study, it is proposed to study the accurate detection of smoke generated when a fire occurs, so the timeliness of fire warnings and the accuracy of fire warnings are better guaranteed in more complex environments.

The researchers in this paper [39], they propose a novel system for detecting fire using Convolutional Neural Networks (CNN). Detection of fire can be extremely difficult using existing methods of smoke sensors installed in the buildings. They are slow and cost-inefficient due to their primitive design and technology. This paper critically ana-

lyzes the scope of artificial intelligence using video from CCTV footage to detect and send alerts. The project uses a built-in dataset containing video frames of fire. The data is then preprocessed and used by the CNN to build a machine learning model. The test set of the dataset is given as input for validating the algorithm and experiments are noted. The project is focused on building a cost-effective and highly accurate device that can be used in almost any fire detection application.
The scope of using video frames in the detection of fire using machine learning is challenging as well as innovative. If this system with a lower error rate can be implemented on a large scale, like in big factories, houses, or forests, it is possible to prevent damage and loss due to random fire accidents by making use of the Surveillance systems. By integrating wireless sensors with CCTV to increase protection and accuracy, the proposed system can be developed into a more advanced system. The algorithm shows great promise in adapting to various environment.

Last and not least this paper [40], detecting forest fires is challenging due to the variety of shapes, textures, and colors they take. Traditional image processing methods depend heavily on human-made extracted features and are not universally appropriate to all forest scenes. To solve this problem, deep learning techniques are used to adaptively learn and extract the characteristics of forest fires. However, Individual learners' limited learning and perception abilities are insufficient to enable them to perform successfully in complex activities. Furthermore, learners tend to pay too much attention to local information and ignore global information, which may lead to false positives.

In this research, a novel ensemble learning strategy for detecting forest fires in various settings is proposed. Firstly, To accomplish the fire detection method, two different learners, EfficientDet and Yolov5, are integrated. Secondly, another single EfficientNet learner is in charge of learning global information to avoid false positives. Finally, the recognition results are based on the decisions of the three learners. Experiments on our dataset show that the proposed method improves detection performance by 2.5% to 10.9%, and decreases false positives by 51.3%, without any extra latency [40].

The successful application of CNN has greatly improved object detection performance. However, a forest fire is an object with no fixed shape that cannot be handled by a single object detector. In addition, fire-like objects easily deceive object detectors, which generate false positives as a result of their limited visual field. To fix all of these problems, this paper proposes a new ensemble learning method for detecting forest fires in real-time. Two powerful object detectors with different expertise (Yolov5 and EfficientDet) are concatenated to make the entire model more powerful for various forest fire scenarios. Then, Introduce a leader (EfficientNet) to lead the detection process and reduce false positives. Experimental results show that this model reaches a superior tradeoff between average precision, false-positive rate, average recall, frame accuracy, and latency. The significant improvements make the model perform well in practical forestry applications.

3.5 Conclusion

In this chapter, we presented the state of the art of fire detection based on deep Learning to improve feature extraction from images. We concluded that we still need to find a robust and high performance method that satisfied our case.

In the next chapter, we are going to talk and document our implementation steps in details, starting from generating our custom dataset to the proposed algorithm that integrates VGG16 transfer learning model with the power of the YOLO architecture that proved itself in the field of object detection.

Chapter 4

Implementation

4.1 Introduction

After talking briefly about Object detection history, the most famous previous research about the fire detection problem, the different approaches used to treat this problem, and finally the results of these studies.
In this chapter, we are ready to get in-depth in the practical part of our project, where we are going to talk about used configuration and hardware, used utils and libraries, and a step-by-step of our implementation phase.

4.2 The Configuration of the Hardware Used

Developing deep learning applications is a heavy computational process it requires a lot of GPU and CPU resources, to implement our solutions we chose to work with the Google collab (cloud) free tier, which comes with two (Intel(R) Xeon(R) CPU @ 2.20GHz) processors, 12.68 GB of RAM and one NVIDIA Tesla T4 GPU.

4.3 Used tools and libraries

4.3.1 Python

When it comes to Data Science and Artificial Intelligence we can't ignore the python programming language, which is an interpreted, high-level, and object-oriented programming language. For these reasons, we have chosen to work with that language.

4.3.2 OpenCV

Since our project is mainly based on Image Processing we have chosen OpenCV for these tasks. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV comes with around 2500 optimization algorithms, including classic and new machine learning algorithms [41].

4.3.3 Selenium

For the automatic generation of our dataset, we decided to implement a scraping workflow that extracts pictures from websites using Selenium Framework. Selenium Framework is an open-source automation tool that is used for task automation, it contains a set of different software tools that makes the automation tasks easy [42].

4.3.4 NumPy

The only way to process images is to use three-dimensional matrices, but working with large matrices will cause a resource issue, especially when using the CPU to run these calculations. To avoid these issues NumPy runs these calculations on the GPU. NumPy is an open-source project aims to enable mathematical computation with Python [43].

4.3.5 TensorFlow

TensorFlow is an open-source machine learning platform that provides a flexible ecosystem of tools used to develop Artificial intelligence and deep learning applications, we have used that platform to generate our models [44].

4.3.6 Keras

Keras is a Python-based deep learning API that runs on the TensorFlow machine learning framework. It was created to allow quick experimentation. The key to successful research is to go as swiftly as possible from idea to results [45]. Keras is:

- **Simple:** but not simplistic. Keras relieves developer cognitive load, allowing you to focus on the most important aspects of the problem [45].
- **Flexible:** Simple processes should be quick and easy, whereas arbitrarily advanced workflows should be possible via a clear path that builds on what you've already learned, according to the Keras principle of progressive disclosure of complexity [45].
- **Powerful:** Keras provides industry-leading performance and scalability, and it's utilized by NASA, YouTube, and Waymo, among others [45].

4.3.7 Jupyter Notebook

Project Jupyter is an open-source, non-profit project that grew from the IPython Project in 2014 to facilitate interactive data science and scientific computing across all programming languages. Using Jupyter made the test phase so easy by interactively changing parameters and functions without rerunning the whole code [46].

4.3.8 Makesense ai

To annotate our dataset images to the YOLOV5 Pytorch TXT format, to do that we have decided to work with Makesense.ai which is free-to-use web tool for labeling photos. The power of that tool is the fact that we don't need to upload our images to their server we will just use them on the client-side.

4.3.9 Unity3D

Is a cross-platform 3D game engine, used widely to develop games for all the devices, the power of unity comes from the ease of use and the good results of graphic, we have used that engine to make a forest simulation.

4.3.10 Django

It is a high-level web framework of python, used to develop simple and complex web applications quickly, we have used that framework to develop our web interface application.

4.4 Dataset generation

As everyone knows that deep learning is a Data-driven approach, which means the more data we collect the more robuste and correct results we get, so in order to give our model more generalization we have tried to collect, clean and categorize our dataset seriously, on the other hand we have decided to only use "Ain Mimoun Forest Fire" related pictures, and these are the steps we have taken:

4.4.1 Collection

Before starting this step we have set our mind on the necessity of collecting more and distinct data, doing that manually would be an unreasonably insane and time-consuming process, so we need to find a way to do that automatically.

After checking various possible sources of images, we have created a python script that uses selenium to scrape google images using a given keyword. the basic idea of that script is to take a given keyword and make a search query on the google pictures search, then loop through every single image and add it to an image list variable, repeating the same process for every single image-related page (algorithm 1).

After running this script on different keywords such as "Ain mimoun feu","khenchela forest snow" , "forest chelia" and "rice aures images" we have successfully generated our first version of our dataset that contains more than 8000 images.

Algorithm 1: Scrapper

Input: keyword:string
Output: images:List of Images

```
images ← []
imagePages ← searchGoogleDriver(keyword)
for (imagePage in imagePages) do
    image ← extractImage(imagePage)
    images.insert(image)
    if image has related images then
        related ← extractRelatedImages(imagePage)
        images.insertAll(related)
return images
```

4.4.2 Video Screenshoter

Making our dataset bigger is good but making it similar and coherent is much more important, this cohesion can be achieved by adding more clean images using other sources rather than scraping, so we made another python script that allow us to take screenshots and extract frames from different downloaded videos, after this step we have just gained another 296 clean images (algorithm 2).

Algorithm 2: Screenshoter

```
Input: video:Video
Output: images:List of Images
video.start()
while video.notEnded() do
    if keyDown = 's' then
        image ← takeImageFrom(video)
        images.insert(image)
    if keyDown = 'q' then
        video.stop()
        break
    if keyDown = 's' then
        video.skipFrames()
        break
return images
```

4.4.3 Cleaning the dataset

As we have mentioned before we have scraped around 8000 images. from the first look, it seems ideal and perfect, but the fact that these images were automatically scraped from google caused a huge amount of repeated images under different names and different sources, so we made another script that detects repeated images by grouping all the images by size then comparing items of each group pixel by pixel using NumPy library and removing the repeated ones (Algorithm 3).

Algorithm 3: Cleaning

```
Input: images:List of Images
imageGroups ← groupImagesBySize(images)
for (group in imageGroups) do
    for (image in group) do
        // remove the repeated image of same group from images list
         repeatedImages ← compareExtractRepeatedOf(group)
         removeReaptedFrom(repeatedImages,images)
return images
```

After running this script we have eliminated up to 4000 repeated images and now we have around 4000 images that need to be verified manually, afterward verifying and cleaning these images manually we got around 2000 clean images ready to be categorized.

4.4.4 Categorizing dataset

After collecting and cleaning our dataset images it needs to be categorized into subfolders based on the class of each image. So we have created another script that shows all the images one by one and reads a keyboard input and moves that image to the corespondent

folder (Algorithm 4).

Algorithm 4: Categorizing

```
Input: imagesFolder:string
images ← readImagesOf(imagesFolder)
for (image in images) do
    cv2.imshow(image)
    key ← cv2.waitKey()
    if(key ==”f”) moveToFolder(image,”fire/fire”)
    else if(key ==”s”) moveToFolder(image,”fire/smoke”)
    else if(key ==”n”) moveToFolder(image,”nofire/nofire”)
    else if(key ==”c”) moveToFolder(image,”nofire/cloud”)
    else show next image
return images
```

By the end of that process we have completed the preparation of our dataset, and now we have two main folders Fire (contains fire and smoke) and No Fire (contains nofire snow and cloudy).

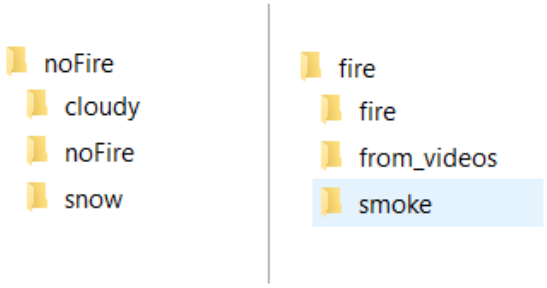

Figure 4.1: First generated dataset folder

4.5 Dataset performance evaluation

In every surveillance system, it’s good to know that there is something wrong, but it’s strongly preferable for that system to give and detect the location of that potential danger, and that one was the major objective of our project, finding the place of fire or smoke in every picture related to forest environments, for that sake we have chosen to make a detection model.

However, starting our project by creating a detection model would be impractical because we haven’t evaluated our dataset from a side and it would be unhelpful for our learning path from another, therefore to test our dataset and to have a good understanding of how deep CNN networks work, we have decided to create a custom CNN model.

4.5.1 Custom CNN architecture

To create our deep CNN network we have used the KERAS Sequential Model class so it will not take a lot of lines to create our model, a lot of tests and changes on the model have been made concerning (deep layers count, learning rate value, epochs number).
All the Models results and issues have been noted and written in the "models result" folder, but the final version of the architecture we have chosen is illustrated in the figure 4.2:

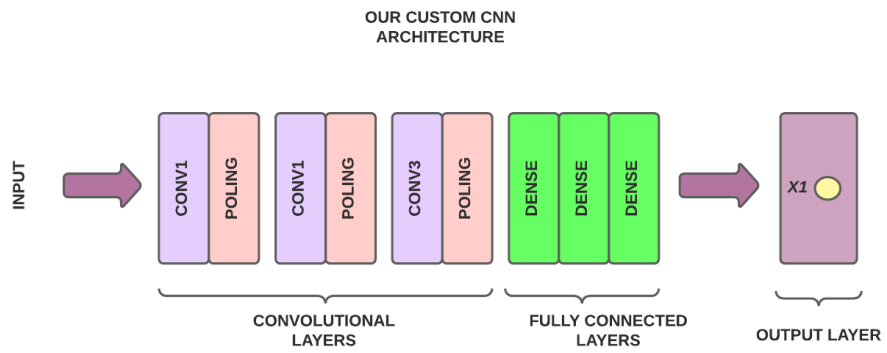

Figure 4.2: Our custom CNN architecture

As it shown in the figure above, our CNN model consists these three main blocks :

- Input Block: this represents the entry point of our model, it receives a 244 by 244 pixels colored RGB image.

- Convolutional Base Block: contains a stack of Conv2D that creates a convolution kernel by taking filters count parameter, followed by a MaxPooling2D layer that downsamples the output of the Conv2D, we have repeated this block three times to get a better features extraction as much as possible.

- Fully Connected Block: feeding the output of "Convolutional Base layers" to a fully connected network to perform classification, that part consists of Flatten layer that unrolls the 3d matrix to a linear input, and three hidden dense layers with different unit counts, followed by Sigmoid activation function.

And this the implementaion of these blocks (Figure 4.3).

```
cnn = tf.keras.models.Sequential()
opt = tf.keras.optimizers.Adam(learning_rate=0.01)
# Input + Convolutional Base Block
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=7, activation='relu', input_shape=[224,224,3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())
# Fully Connected Block
cnn.add(tf.keras.layers.Dense(units=160, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=180, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=64, activation='relu'))
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Figure 4.3: Custom CNN Model

4.5.2 Dataset loading

After choosing and implementing the right architecture, we only need to load our dataset images and resize them to fit the input shape of our CNN, this can be done manually by loading all the pictures one by one and resizing them but this will cause Memory saturation problems.

Fortunately, Keras provides a set of useful utils one of them is the ”image dataset from directory” function which takes the images directory folder and returns a ”tf.data.Dataset” object that yields batches of images from the subdirectories ”Fire” and ”NoFire” during the training and validation phases so it will save us a lot of memory ressources, it also can be used to split resize and normalize our dataset, before running this function the provided directory should contain two folders the first one that holds images for fire class and the other for the no fire class, and we are all set to go and train our model.

4.5.3 Experimental results

Training deep CNN networks is a heavy computational process so it needs a lot of GPU or CPU resources. So at that point, we have decided to train all of our models using the Google Collab since the free pack account is enough for our models, even though our CNN model contains a lot of dense layers it only took half of hour to complete the 30 epochs training and these were was the results (table 4.1):

Epoch	Accuracy	Validation Accuracy
28	100%	97%
29	100%	97%
30	100%	98%
Best	99.4%	98.44%

Table 4.1: Custom CNN model results

As the table above shows, we got a well validation accuracy of 98% for the best epoch. So technically speaking, we are sure that images are usable and valuable.

However, there are some issues, the quality of our pictures is so weak when it comes to the smoke particles and this will make a huge problem in the detection phase. From another side, our goal isn't just to tell if there is a fire or smoke, our application should have the power to at least give the approximate location of smoke and fire.

Additionally, due to the small number of images in our dataset, the features extraction part of our model (CNN) feels weak, to fix this problem we can augment the number of epochs to a huge number but this is hard to implement due to the lack of GPU resources even using the google collab it would take a days or weeks to achive that goal.

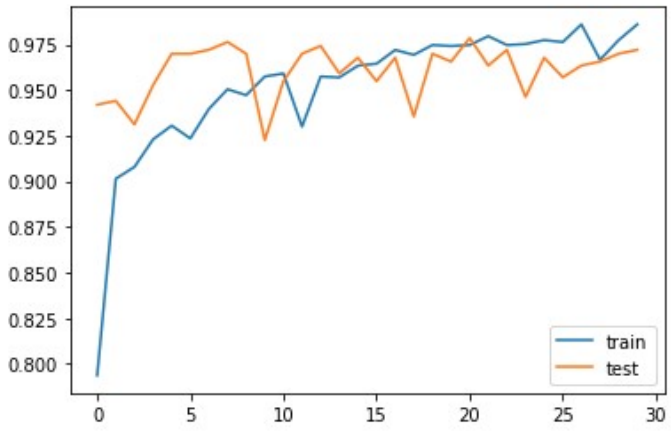

Figure 4.4: CNN history curve.

4.6 Detection model

When it comes to the most popular detection CNN models there is a plenty of choices, we can't ignore the R-CNN family and YOLO algorithms, these two algorithms have made a giant leap in the object detection computer vision tasks.

It's been challenging to choose between the Faster R-CNN and YOLOV5 since the two are the most used for detection problems. But after a long time of the search, we have found that the YOLOV5 model provides a significant advantage in detection speed where the frames per second (FPS) was more than eight times than Faster R-CNN [47] and that quite perfect for our project.

4.6.1 YOLOV5 dataset preparation

Since YOLO is a supervised learning method, it is necessary to provide labeling information for each image we fed to the model, every version of YOLO accepts a different dataset format, and as we are using the YOLOV5 we have to convert our dataset to "YOLOv5 PyTorch TXT Annotation Format" [48], For every single image we have to generate a corresponding text file that contains many lines, each line represents a bounding box as an example for the next image (see Figure 4.5).


```
0 0.827241 0.274059 0.170991 0.391632
0 0.420991 0.454812 0.266509 0.405809
```

Figure 4.5: Our dataset formatted for YOLO.

In the figure 4.5 there are 2 objects in total (fire), each line represents one of these objects. The specification for each line should follow these roles: [0]

- each row represents an object.
- each row contains class number, coordinates of the bounding box x-center y-center and width height.
- box coordinates are normalized dependent on the dimensions of the image (between 0 and 1).
- class numbers are zero-indexed (start from 0).

that was a time-consuming process, since we have labeled every single image from our dataset images manually, we have done that using the "Make sense AI" labeling web tool (see Figure 4.6), without mentioning the quality problem of smoke images of our dataset at that point we decided to use same of public free smoke images to fix that issue [49].

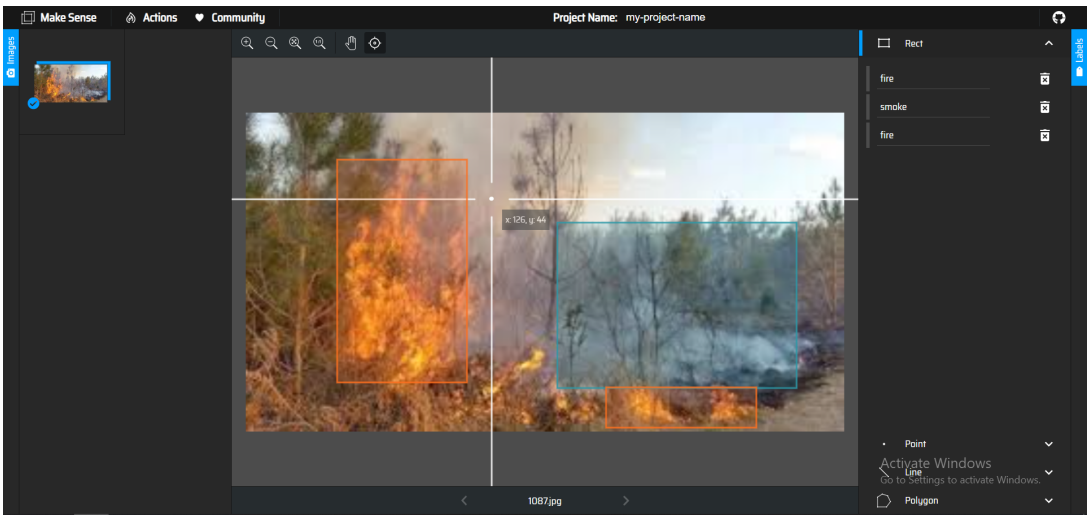

Figure 4.6: Make sense AI labeling tool.

4.6.2 YOLOV5 detection model results

We took a lot of time training YOLOV5 models to find the perfect version that gives more accuracy, so we have tried a lot of images combination, modifying labels, removing low-quality images (more than 40 trained models), we have ended up training YOLOV5 with 1800 carefully selected images fine-tuned with coco pre-trained weights for 100 epochs and these are the results (table 4.2):

Class	Accuracy	Validation Accuracy
All	96.3%	95.3%
Fire	92.6%	91.4%
Smoke	100%	99.2%

Table 4.2: YOLOV5 detection results

As the table above shows, we got good results for a custom generated dataset, the validation accuracy around 95% and that's quite perfect, and these are some of the images predicted by the model (figure 4.7):

Figure 4.7: Dataset predicted by YOLOV5

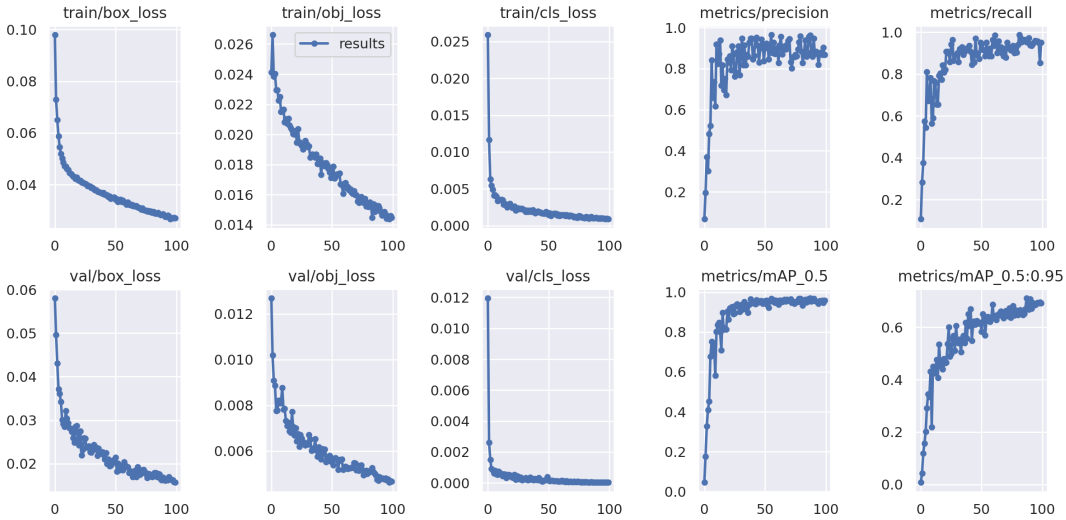

Figure 4.8: YOLOV5 training metrics

That was the bright side of our detection model. Unfortunately, due to the lack of high-quality images of "no fire" in our dataset our model does some imprecise detection, sometimes it detects stones and dirt as fire, sometimes blue sky as smoke (figure 4.9).

(a) Stone false detection

(b) Sky false detection

Figure 4.9: YOLOV5 false detections cases

On the other hand, there is an issue concerning GPU resources even using google collab each frame detection takes 0.02 to 0.03 seconds to complete the detection process and extensively much longer than our custom CNN classification model, to illustrate that problem, let's suppose that case a 24/7 camera connected to our detection system, and as we all know that in a normal case in a forest environment there is no fire or smoke, so running our detector model on every frame of every second for the whole day would be an increasngly resources consuming process, and that's one of the downsides of using YOLOV5 on our case, so we have to find a solution to these problems and make our model perform in 24/7 real world projects.

Figure 4.10: Dataset quality issue

4.7 Proposed solution

Before going through our proposed solution, let's define some concepts that we have used:

- Transfer Learning: refers to techniques for transferring knowledge from multiple sources to other problems to be solved. This can be implemented by using pre-trained model and freeze some layers, and do additional training to the rest of layers to adjust for your purposes and goals.
- Fine tuning: fine tuning of the network is the process when you adjust parameters such as a learning rate, the number of epochs, the optimizer, regularization parameters etc. to achieve best possible results.
- Ensemble learning: is the technique of using multiple algorithms or models to obtain better results and performance than using one algorithm.

To fix the issues with our detection model, we have tried a lot of ideas that can improve the results, from increasing the epochs number and changing the coco weights used for the fine tunings, filtering the images of our dataset, and removing those who can cause false detections, and it didn't go very well without mentioning the resources issue.

That was the point where we got pretty sure that we have got the best possible results from the YOLO detection model, and we need to solve these issues with different approaches, so for the high resource usage issue, running the detection process on every frame of our videos would be an unreasonably insane process, we only need to run the detection process when we get a sign that there's a kind of danger whether it's fire or smoke.

So we have decided to make a new algorithm that finds whether an image may contain a fire or smoke if the image is clean, the result would be marked as save, else the algorithm will pass the image to the YOLOV5 detection model and returns the location of the fire or smoke in the image. So technically speaking, our algorithm will contain two main parts which are:

- Evaluation model: that part is responsible for the evaluation process of an image, it takes an image and returns a score between 0 and 1 that represents how dangerous the image is, if the score is less than 0.5 the image will be marked as SAVE, else the algorithm will pass the image to the second part (the detector), in other words it's just a CNN network that evaluate images.

- Detector: that part will detect fire and smoke objects of an image only if the evaluation model marked the image as a dangerous image, and this is our YOLOV5 model, so it has already implemented.

Algorithm 5: Proposed Algorithm

```
Input: frame:Image
evaluationScore ← evaluateImage(frame)
if evaluationScore < 0.5 then
    // Score < 0.5, there is no fire , mark image as save
    result ← markAsSave(frame)
else
    // detect and draw Fire or smoke locations
    result ← yoloDetector(frame)
    // Mark image as damger
    result ← markAsDanger(result)
return images
```

4.7.1 The evaluation model

The main idea was to utilize our custom CNN model used in the dataset performance evaluation step, but since the issues concerning the power of features extraction addressed above needs a lot of resources and time to be fixed, Imaging training the model for days or weeks just to make convolutional base performs better, that would be an increasingly time-wasting process, without mentioning the lack of GPU resources to implement that.

So instead of training a new model from scratch, we have made reachers of things we can do. After a brief search, we found the ImageNet ILSVRC challenge that aims to improve and develop new models that can classify a million (more than 14 million images) pictures into thousands of classes (more than 22 thousand) [50], thanks to that challenge there is a good number of model architectures that can perform very well on large datasets one of these is the VGG16 architecture.

VGG16 is a deep CNN network architecture trained on a subset of the ImageNet dataset

to get the best results for that challenge, the training phase took 2–3 weeks using a system equipped with Four NVIDIA Titan Black GPUs [51], the model achieved a 92.7% classification accuracy which is perfect, but what serves our goal is the fact that VGG16 is a public pre-trained architecture that can extract generic visual features in most of the cases whatever the dataset size.

Since the VGG16 have been trained with a huge amount of images (ImageNet) for a long time, the convolutional base of that model would have a strength at feature extractaction and that will help us fix the issue of feature extraction we have encountred before.

As we only intrested the power of feature extractaction in the VGG16 model we only want to use his Convolutional base, we don't need to import the whole parts of the architecture due to the huge number of parameters and size of the fully connected layers, and these are the steps we have taken:

1. Importing and downloading the pre-trained VGG16 with saved weights and biases.
2. Removing the dense layers of the network and just leaving the convolutional base.
3. Appending our custom fully connected layers to the imported convolutional base.
4. Applying Fine-Tuning approach by freezing the imported convolutional base so they won't be trained during the training phase.

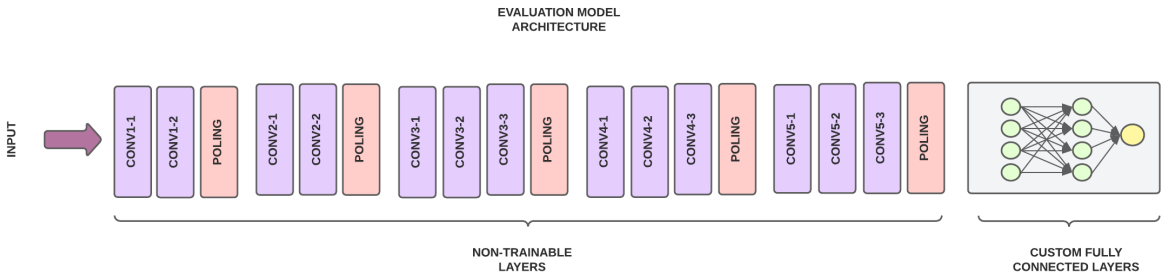

Figure 4.11: Evaluation model architecture

freezing the imported convolutional base will prevent these layers from updating during the training since they already trained before and this will improve the performance of the whole architecture by reducing the number of trainable parameters (>14 millions reduced parameters) on figure 4.12.

```
=================================================================
Total params: 16,829,641
Trainable params: 2,114,953
Non-trainable params: 14,714,688
```

Figure 4.12: Evaluation model trainable parameters

After all, we are ready to train the final proposed model with our generated dataset, only difference here that we have splitted our dataset into two sub-folders, the first one that represent the dangerously images (contains fire and smoke images) , and the other for clean and save images.

4.7.1.1 Traning results

After training the resulted model for 30 epochs on the google collab, the first thing we have noted is the training was so quick compared to the original VGG16 network thanks to the freezing step. it only took about one hour to complete the 30 epochs training and the results were so promising (table 4.3):

Epoch	Accuracy	Validation Accuracy
28	98.57%	100%
29	98.43%	100%
30	98.48%	98.6%
Best	98.57%	100%

Table 4.3: Evaluation model result

As the table above shows, we got a promising and perfect validation accuracy of 100% for the best epoch, using the same old dataset used in the first custom CNN Model. Compared to all of the previous models we made this is the first time we got a 100% for validation accuracy for the whole dataset with just 30 epochs and an hour of training, and for sure we will get a better and stable results if we just increased that number.

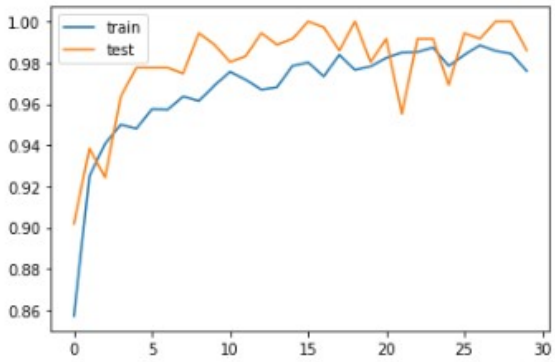

Figure 4.13: Training and validation curves of our custom VGG16.

However, to give these numbers a sense of real time performance we have tested the model with the worst quality and most ambiguous pictures we have in our dataset and these were the results (figure 4.14): Looking to these results it is clear that our evaluation model is ready to go, and the only thing left to do is to implement the proposed algorithm by integrationg the evaluation model and the YOLOV5 detector to create the final model.

Figure 4.14: Evaluation model results

4.7.2 Final architecture

After completing the two main parts we have explained above, and just to recall, the algorithm will take an image as an input, feed it to the evaluation model, and verify if that result is greater than 0.5 it will run the YOLOV5 detector, else it will mark the image as clean, using that approach will save us a lot of GPU ressources as shown before, and will detect even the low quality images thanks to the evaluation model. we only need to use them on the way explained on the (figure 4.15):

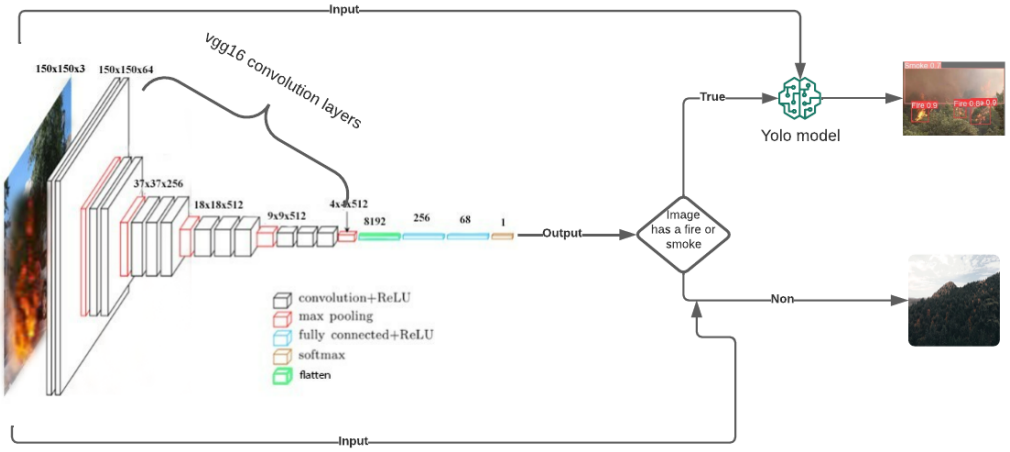

Figure 4.15: Proposed algorithm architecture

4.7.3 Final results

The main goal of our project was to be executed on live video cameras or drones as video sources, but due to a lack of time and resources we couldn't afford that, but to avoid these problems and to test our final model on a real-world scenario that matches our needs, we have tested it on a public youtube video,of a place that shares the same geographic features of "Ain Mimoun Forest" (original video [52]), these are some the snapshoot token from the result of our proposed model in the figure 4.16 (full video result can be found in the attachements):

Figure 4.16: Video results snapshoot 1

From the figure above we can conclude that the issue of wrong detection has gone (stone problem mentioned on figure 4.9a), if we take a deep look we can see the danger sign on the top left corner that's the result of the evaluation model, another example to see is the case of fire or smoke (figure 4.17) where we can see a 100% for the danger sign:

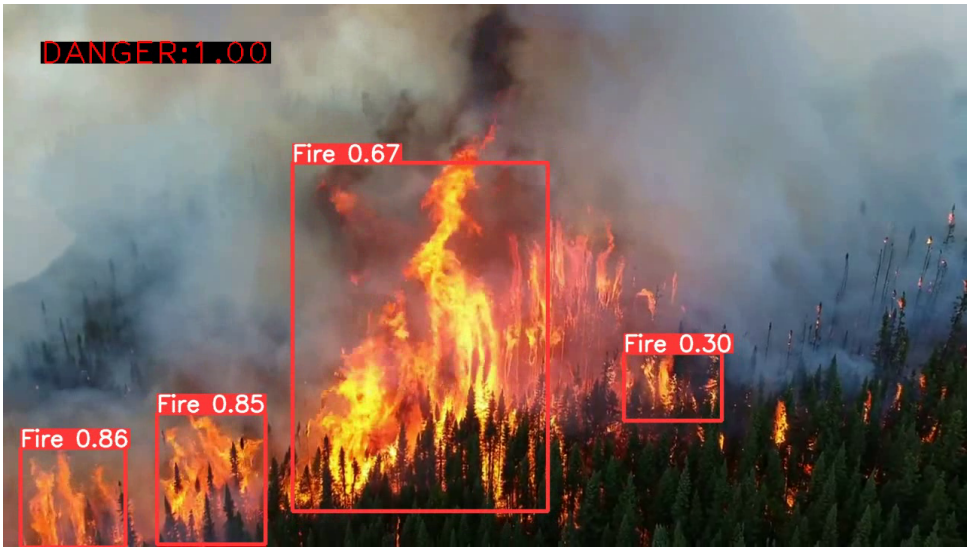

Figure 4.17: Video results snapshoot 2

4.8 Simulation with Unity3d

After completing the implementation of our proposed algorithm, and testing it on a drone-taken video, to make sure that our result can be used for a real-world case we have decided to test our video on a forest environment simulation, so we used the Unity3d game engine to create a custom forest environment where we have tried to mimic the Ain Mimoune forest region as much as possible, the created scene contains trees, grasses, and bushes that are similar to these of Khenchela region (figure 4.18).

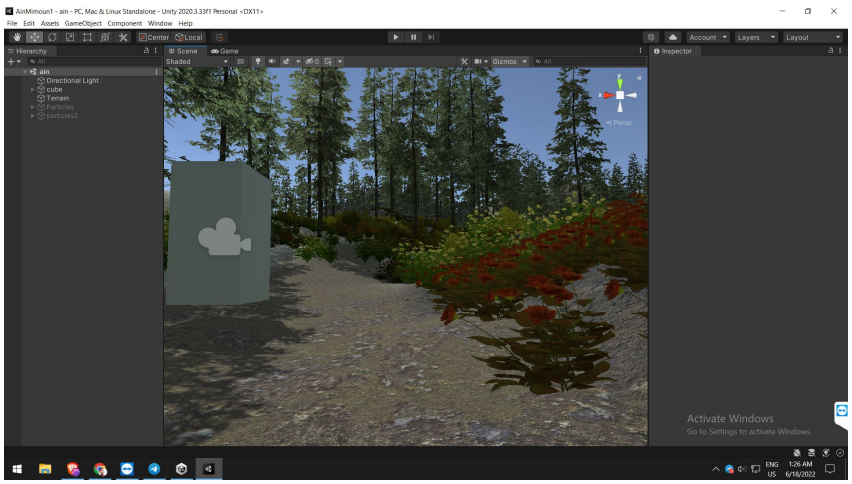

Figure 4.18: Unity3d forest environment

After completing the environment we have used a custom C script that generates fire and smoke particles around the world for the sake of adding some reality, now we have high-quality forest pictures with and without fire and smoke, after that we made another script that takes screenshots from Unity3d inside the 3d world.

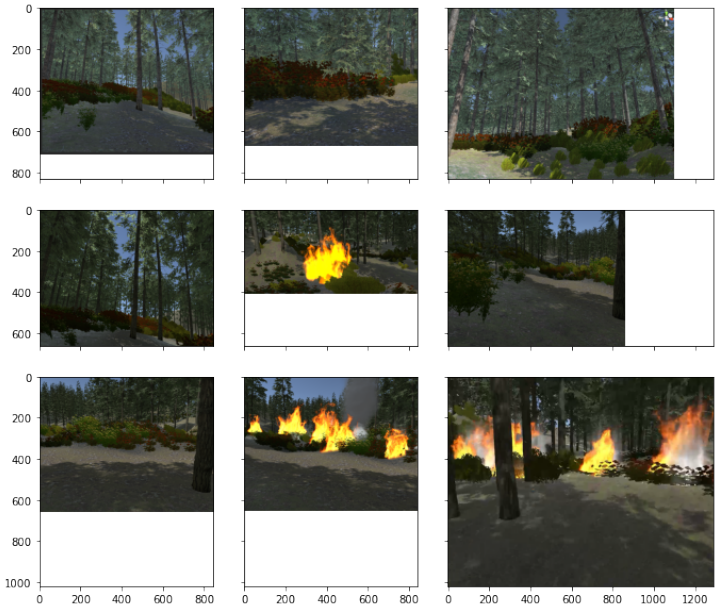

Figure 4.19: Unity3d input pictures

After taking some screenshots and a video recording, we have fed these data to the proposed algorithm, and these were the results (figure 4.18).:

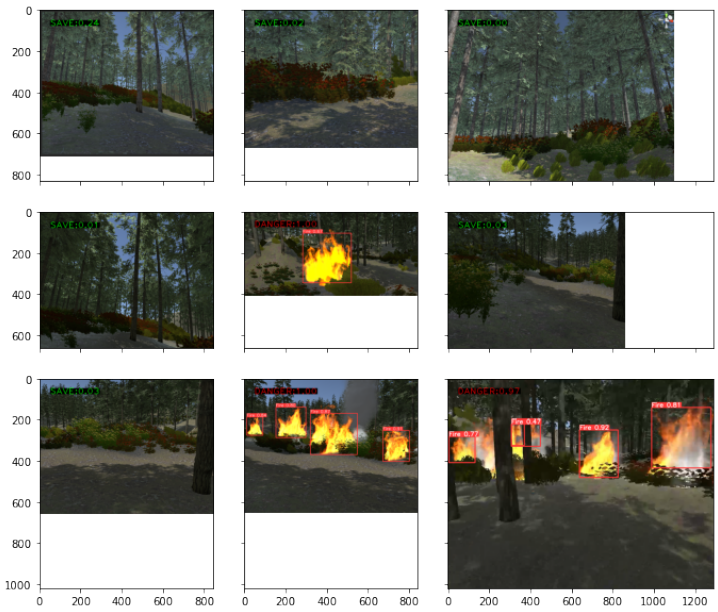

Figure 4.20: Unity3d predictions result

Using the simulation step makes it easy to generate burned forest trees cause in real-world we can't generate burned trees pictures unless we burn the whole forest, or to simulate the fire and smoke spreading pattern to find a solution based on that, also we can use this environment to generate high-quality dataset images that can be used on the same models we stated above.

4.9 Web application

We have built a custom server to treat video streams as we can see in the figure bellow:

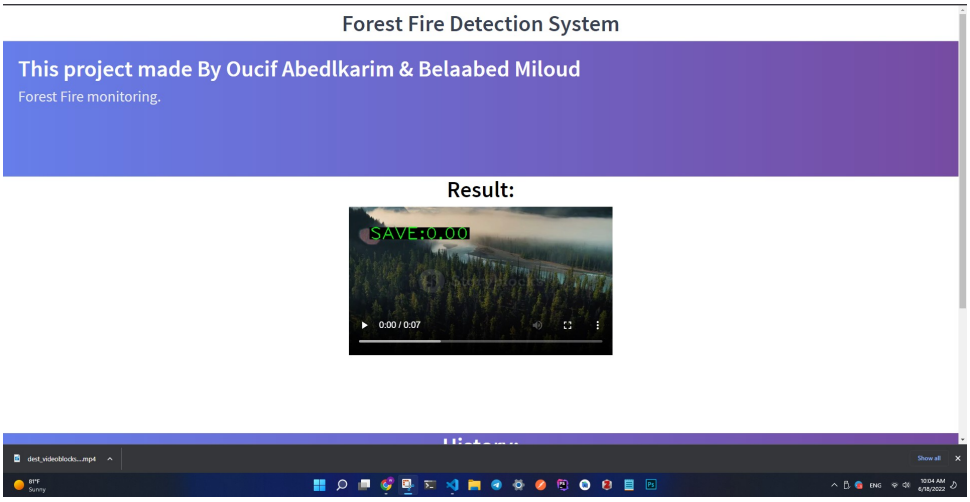

Figure 4.21: Web application

We created this web app based on the Django framework and SQLite3 database. We chose the MVC pattern, which is a software architecture pattern, that is based on three parts:

- **Model:** represents the data and logic of the app in which our object occurs.

- **View:** the view that the final user is going to interact with.

- **Controller:** the controller is the go-between for models and views. It relays data from browser to app and from app to browser. It contains the business logic on which our system is based.

Why have we chosen this architecture? Because it is simple, and we don't have a large-scale application to use other complex architectures, such as the clean architecture that was released by Uncle Bob in 2012 or the micro-services architecture.
What's our purpose for building this web app and deploying it? We want to make our work public, and we're making it in this way just to make it easy for the non-computer science users to test it and collect fire forest videos as much as possible.
Our feature plans are: After collecting more videos and getting a deep understanding of our system, we going to deliver web services for the drones to use our system directly. We are going to monitor everything from this web app, as much as possible, from any device, such as smart phones,etc. Here is a usecases diagram to see the interaction of our system with the actors.

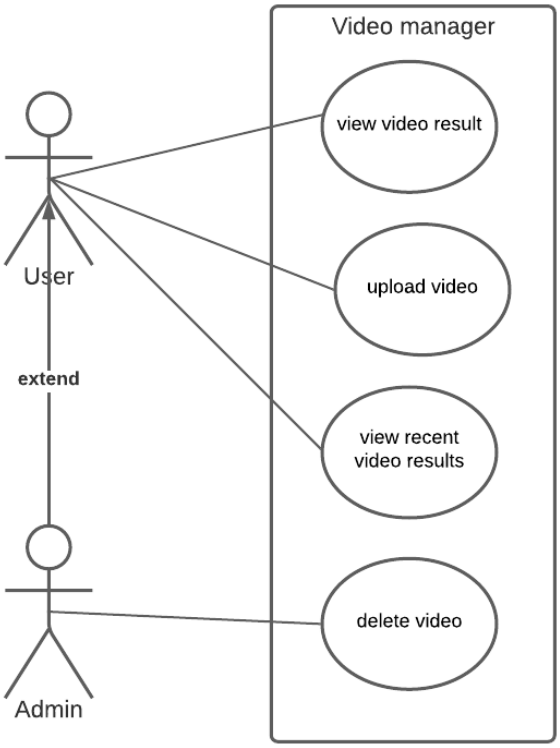

Figure 4.22: Use-cases diagram

4.10 Limits

After completing our implementation part in detail, the steps, research we did, and limits we encountered to accomplish our final proposed algorithms, we are going to state some of the main limits we had struggled with.

To start, training CNN models for forest environments needs a lot of coherent and high-quality images to avoid false classifications, certain cases should take into consideration such as the similarity between fire smokes and clouds, that case has been treated partially on our dataset, but for real-world projects, it needs a lot of images, the same goes to the sunset and fire similarity, the same thing for YOLO detection model.

However, after trying different approaches to solve the main issue, we can't just detect fire or smoke using two-dimensional arrays of colors, it is clear that efficient early fire detection systems requires dimensions and parameters must be taken into consideration such as the thermal map pictures and air pressure measurements, with these extended data, we can create a valid and strong system that can use to detect and segment smoke and fire precisely instead of just detecting using a bounding box, which appears to be the best approach from our point of view, we can let that approach for our future searches.

For the simulation step, due to the lack of our resources GPU and CPU, it was so hard to make the quality of the simulated environment close to the real one, but if we have provided a GPU for that we can make a high-quality and real close dataset images.

4.11 Conclusion

In conclusion, we have generated our dataset using custom algorithms (collect, clean, and categorize), evaluate the quality of these images using a CNN model, made a YOLOV5 detection model, and our final proposed algorithm that merges between VGG16 transferred training model and YOLOV5, testing our solution on a simulated environment made us sure that we came to an end solution that gave us a good result for our dataset and of course some of the suggestions that can be used for future works concerning forest fire problem.

4.12 Conclusion

Conclusion and perspectives

General conclusion

In recent years, mainly due to advances in deep learning, the quality of image description, and object detection have improved at a breathtaking pace. Much of this progress is the result of more powerful hardware, larger databases (DB), and larger models and a consequence of new ideas, improved algorithms, and network architectures. « Thanks to the Deep Learning, the future of artificial intelligence is promising » .

One of the main computer vision applications based on the technique of machine learning is object detection. For many years, scientists have looked for effective ways to achieve good results in this area. Methods such as that of Viola and Jones or the HOG have been the reference for several years. They allowed a major turning point in the object detection problem. But in 2012, the marriage of deep learning with this discipline dethrones them and highlights CNN.
This led us, in this thesis, towards the development of the detection system object as a mechanism to help detect fire and smoke in a large area of forest, because of what we have seen in the last year (the mountainous Kabylie region fire disaster, and the forest of ain Timimoun disaster).

To realize that system, the first step was to generate a fully custom images dataset of "Khenchela Ain Mimoun forests pictures" using custom algorithms, evaluate the quality of these images with a custom CNN to make sure they are reliable, creating our first detection model based on the YOLOV5 architecture that has proven its self in the object detection for real-time projects, discussing the results we got that can break our system, and finally, propose a fully custom algorithm based on the VGG16 pre-trained model that we have changed its structure by implementing the transfer learning approaches to address the issue of dataset limitations, the same algorithm integrates the YOLOV5 model with the VGG16 by implementing another approach "Ensemble learning".

In conclusion, the results of our proposed algorithm gave us extraordinary detection results from different sides, from performance and accuracy, resources usage, to time complexity, so we can say it was an Efficient system.

However, there is always a place for improvement even if our system was quite good for our case and dataset, it can be improved using other approaches to solve that problem for example use segmentation instead of detection with Masked-RCNN that can give huge and perfect results instead of drawing a bounding box it will perform a pixel-level fire detection. On another side, using high-quality images will give the system a huge boost toward the real-time projects, especially for the YOLOV5 detector part which needs some improvements, without mentioning the option of changing the YOLOV5 backbone that can change the detection process results.

Perspectives

Finally, from our perspective, early fire detection systems can't just depend on two-dimensional arrays of colors, it is clear that a perfectly efficient system requires other dimensions and parameters that must be taken into consideration such as the thermal map pictures and air pressure measurements, using these data our system will be perfect.

Another thing, as we know the most challenging thing here in our country is collecting the dataset, for the sake of getting a better and large amount of dataset we were thinking to create an environment using one of the most popular tools for game simulation Unity3D, it gives us the opportunity of generating a new dataset based on the environment that we have created, and the accessibility to test our system on it, but sadly, we couldn't implement it in practice because of time and it needs high quality of hardware to prepare a good quality resolution of the environment.

Bibliography

[1] A. Thind, *A Brief Introduction to Artificial Intelligence*, en, Nov. 2021. [Online]. Available: https://medium.com/@arpanthind/a-brief-introduction-to-artificial-intelligence-c556e7dab0f4 (visited on 03/24/2022).

[2] *A Complete History of Artificial Intelligence*, en-us. [Online]. Available: https://www.g2.com/articles/history-of-artificial-intelligence (visited on 03/24/2022).

[3] T. R. Kurfess, Ed., *Robotics and automation handbook*, en. Boca Raton, Fla.: CRC Press, 2005, ISBN: 978-0-8493-1804-7.

[4] P. Prasad, "Machine learning process on social media," Aug. 2020. DOI: 10.26524/jms.2020.

[5] R.K, *« Machine learning » : Apprentissage supervisé ou non supervisé*, en, Mar. 2018. [Online]. Available: https://medium.com/@bobkrc/machine-learning-apprentissage-supervis%C3%A9-ou-non-supervis%C3%A9-bced5be4fd7f (visited on 03/29/2022).

[6] V. Maini and S. Sabri, "Machine Learning for Humans," en, p. 97,

[7] B. Mahesh, *Machine Learning Algorithms -A Review*. Jan. 2019. DOI: 10.21275/ART20203995.

[8] K.-L. Du and M. Swamy, "Reinforcement Learning," in, Dec. 2014, pp. 547–561, ISBN: 978-1-4471-5570-6. DOI: 10.1007/978-1-4471-5571-3_18.

[9] L. Salsabil and S. Amaria, "Présenté et soutenu publiquement par," fr, p. 63,

[10] luisquintanilla, *Tâches d'apprentissage automatique - ML.NET*, fr-fr. [Online]. Available: https://docs.microsoft.com/fr-fr/dotnet/machine-learning/resources/tasks (visited on 04/03/2022).

[11] B. Rokad, *Machine Learning Approaches and Its Applications*, en, Feb. 2020. [Online]. Available: https://medium.datadriveninvestor.com/machine-learning-approaches-and-its-applications-7bfbe782f4a8 (visited on 04/05/2022).

[12] ENGINEERING TUTORIAL, *Biological vs Artificial Neural Networks | A Comparison | Neural Network*, 2020. [Online]. Available: https://www.youtube.com/watch?v=Mu1o4hT6G1E (visited on 04/05/2022).

[13] J. P. Adam Gibson, *Deep Learning A Practitioner's Approach*, First Edition. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., Jul. 2017, ISBN: 978-1-4919-1425-0.

[14] D. Karunakaran, *Deep learning series 1: Intro to deep learning*, en, Oct. 2019. [Online]. Available: https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20 (visited on 04/04/2022).

[15] *Automatic Log Analysis using Deep Learning and AI*, en. [Online]. Available: https://www.xenonstack.com/blog/log-analytics-deep-machine-learning (visited on 04/04/2022).

[16] A. Mohanty, *Multi layer Perceptron (MLP) Models on Real World Banking Data*, en, May 2019. [Online]. Available: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f (visited on 04/05/2022).

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/5.726791.

[18] C. Camacho, *Convolutional Neural Networks*. [Online]. Available: https://cezannec.github.io/Convolutional_Neural_Networks/ (visited on 04/08/2022).

[19] *How do computers see an image? | CommonLounge*. [Online]. Available: https://www.commonlounge.com/discussion/244616b76d3d40f88e8f12103a22743d (visited on 04/09/2022).

[20] D. H. C. Pérez, "A practical approach to Convolutional Neural Networks," en, p. 70,

[21] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, en, Dec. 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 04/09/2022).

[22] A. Deshpande, *A Beginner's Guide To Understanding Convolutional Neural Networks*. [Online]. Available: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (visited on 04/10/2022).

[23] *4. Fully Connected Deep Networks - TensorFlow for Deep Learning [Book]*, en, ISBN: 9781491980453. [Online]. Available: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html (visited on 04/11/2022).

[24] *Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html (visited on 04/11/2022).

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv:1311.2524 [cs]*, Oct. 2014, arXiv: 1311.2524.

[26] B. Wang, "Research on Pedestrian Detection Algorithm Based on Image," *Journal of Physics: Conference Series*, vol. 1345, p. 062 023, Nov. 2019. DOI: 10.1088/1742-6596/1345/6/062023.

[27] R. Girshick, "Fast R-CNN," *arXiv:1504.08083 [cs]*, Sep. 2015, arXiv: 1504.08083.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jan. 2016, arXiv: 1506.01497.

[29] U. Handalage and L. Kuganandamurthy, *Real-Time Object Detection Using YOLO: A Review*. May 2021. DOI: 10.13140/RG.2.2.24367.66723.

[30] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv, Tech. Rep. arXiv:2004.10934, Apr. 2020, arXiv:2004.10934 [cs, eess] type: article.

[31] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," arXiv, Tech. Rep. arXiv:1911.11929, Nov. 2019, arXiv:1911.11929 [cs] type: article.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in, vol. 8691, arXiv:1406.4729 [cs], 2014, pp. 346–361. DOI: 10.1007/978-3-319-10578-9_23.

[33] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," arXiv, Tech. Rep. arXiv:1803.01534, Sep. 2018, arXiv:1803.01534 [cs] type: article.

[34] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv, Tech. Rep. arXiv:1804.02767, Apr. 2018, arXiv:1804.02767 [cs] type: article.

[35] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," *arXiv:1905.05055 [cs]*, May 2019, arXiv: 1905.05055.

[36] *Image fire detection algorithms based on convolutional neural networks | Elsevier Enhanced Reader*, en. DOI: 10.1016/j.csite.2020.100625. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S2214157X2030085X (visited on 04/13/2022).

[37] A. Namozov and Y. I. Cho, "An Efficient Deep Learning Algorithm for Fire and Smoke Detection with Limited Data," en, *Advances in Electrical and Computer Engineering*, vol. 18, no. 4, pp. 121–128, 2018, ISSN: 1582-7445, 1844-7600. DOI: 10.4316/AECE.2018.04015.

[38] B. Zhang, L. Sun, Y. Song, W. Shao, Y. Guo, F. Yuan, 1 School of Computer Science and Technology, Shandong Technology and Business University, Yantai, Shandong 264005, China, 2 Fire Rescue Brigade, Yantai Hi-tech Industry Development Zone, Yantai, Shandong 264670, China, and 3 Plaza Park Management Service Center, Yantai Laishan Zone, Yantai, Shandong 264670, China, "DeepFireNet: A real-time video fire detection method based on multi-feature fusion," en, *Mathematical Biosciences and Engineering*, vol. 17, no. 6, pp. 7804–7818, 2020, ISSN: 1551-0018. DOI: 10.3934/mbe.2020397.

[39] G. Vadakkadathu Rajan and S. Paul, "FOREST FIRE DETECTION USING MACHINE LEARNING," Jan. 2022.

[40] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A Forest Fire Detection System Based on Ensemble Learning," en, *Forests*, vol. 12, no. 2, p. 217, Feb. 2021, Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1999-4907. DOI: 10.3390/f12020217.

[41] *About*, en-US. [Online]. Available: https://opencv.org/about/ (visited on 04/21/2022).

[42] Fei Wang and Wencai Du, "A Test Automation Framework Based on WEB," en, in *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, Shanghai: IEEE, May 2012, pp. 683–687, ISBN: 978-1-4673-1536-4. DOI: 10.1109/ICIS.2012.21.

[43] *NumPy*. [Online]. Available: https://numpy.org/about/ (visited on 05/31/2022).

[44] *TensorFlow*. [Online]. Available: https://www.tensorflow.org/?hl=fr (visited on 04/21/2022).

[45] K. Team, *Keras documentation: About Keras*, en. [Online]. Available: https://keras.io/about/ (visited on 04/21/2022).

[46] *Project Jupyter*, en. [Online]. Available: https://jupyter.org (visited on 05/26/2022).

[47] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification," en, In Review, preprint, Jul. 2021. DOI: 10.21203/rs.3.rs-668895/v1.

[48] *How to Train YOLO v5 on a Custom Dataset*, en, May 2021. [Online]. Available: https://blog.paperspace.com/train-yolov5-custom-data/ (visited on 06/04/2022).

[49] *Wildfire Smoke Dataset*. [Online]. Available: https://public.roboflow.com/object-detection/wildfire-smoke (visited on 06/10/2022).

[50] 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," en, *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y.

[51] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, Tech. Rep. arXiv:1409.1556, Apr. 2015, arXiv:1409.1556 [cs] type: article. DOI: 10.48550/arXiv.1409.1556.

[52] Sanuck176, *DJI Spark flies over forest fire.* Feb. 2019. [Online]. Available: https://www.youtube.com/watch?v=M97sJdyeEM4 (visited on 06/08/2022).