PEOPLE'S DEMOCRATIC REPUBLIC OFALGERIA
Ministry of Higher Education and Scientific Research
Mohamed Khider University – BISKRA
Faculty of Exact Sciences, Natural Sciences and Life
Computer Science Department

Order N°: IA_StartUp 03 /M2/2023

Thesis

Presented to obtain the academic master's degree in

Computer Science

Option: **Artificial Intelligence (AI)**

Safe desert houses

By :

BEN BRAHIM RAIANE

BEN HAMZA MARWA

Defended on _/07/2022 in front of the jury composed of:

ALOUI Ahmed	MCA	President
SAHRAOUI Somia	MCA	Supervisor
MERIZIG Abdelhak	MCB	Examiner

Academic year 2022-2023

Acknowledgements

First and foremost, we would like to express our sincerest gratitude to the Almighty ”ALLAH” for bestowing upon us faith, health, strength, courage, patience, and perseverance, and for granting us the resources to complete this humble endeavor. We extend our heartfelt thanks to ”ALLAH” for illuminating our path towards success.

We would also like to convey our appreciation to Prof. SAHRAOUI Somaia, our esteemed supervisor, for her willingness to guide us and for her unwavering patience and invaluable advice. With dedication and genuine interest, she steered this project with enthusiasm, generously sharing her scientific expertise. We extend our deepest gratitude to her, along with our profound admiration for the attention she devoted to this dissertation, her constant encouragement, trust, unfailing availability throughout its completion, and her kindness towards us.

Once again, we extend our warmest thanks to Okba AL Annabi for his timely assistance, invaluable advice, and humble demeanor. We are truly grateful for his contribution to our journey, and we deeply appreciate his unwavering support.

Furthermore, we would like to express our gratitude to each member of the jury, for graciously accepting the responsibility of evaluating our modest work. May they find within these words our sincere appreciation and utmost respect.

We would like to express our heartfelt thanks to all our friends and colleagues who have provided us with support throughout our research journey. Their assistance, both direct and indirect, has been invaluable in helping us successfully complete our research project.

Benhamza Marwa

This project is wholeheartedly dedicated to our beloved family:

Our fathers: Ahmed, Ali

Our mothers: BENHAMZA Dalila, CHAIBAI Djahida

For their encouragement, their dedication, their sacrifices for our happiness and our success, their love, and their wisdom which allowed us to reach rank

No dedication can express my great admiration, consideration, and sincere affection for you.

No matter what we do, we could never reward you for the great sacrifices you have made and continue to make forus.

May ”Allah” protect you and grant you a long life full of health and happiness

Our sisters: Samira, Randa, Fatima el Zahra , Malek,

Our Brothers: Mahdi, Mohamed El Taher,Rabeh Houssem el Dine

For their constant presence, their sympathy, their support, and their encouragement they gave me. My pride to whom I wish success.

To everyone who ever supported us in our life...

Abstract

Algeria's deserts areas (Sahara) are known (characterized) by the presence of poisonous animals of varying severity (scorpions and snakes), that choose difficult places to hide what makes it hard for us to see and find them ,as a result the high number (percentage) of poisoned people especially children and aged ones . Usually traditional techniques are used to detect and catch poisonous animals, we search and kill them whenever we see them, also we find people of the areas where such animals exist grow some kinds of animals that have the ability to search and find the poisonous animals and get rid of them like cats and chickens.

This project aims to detect snakes and scorpions in residential areas where these animals are present, as well as to provide timely and immediate warnings about their presence. The goal is to prevent and protect the people of these areas from being exposed to venomous bites, which can lead to serious symptoms and, in some cases, even death.

The presented work is a smart snakes and scorpions detection system.This system consists of two main devices, a camera to detect poisonous animals, and the homeowner's phone, to send and receive alarm and messages notifications between the camera and the homeowner. The system is developed using the deep learning and computer vision techniques. We used YOLOv8 model to train our model for the snakes and scorpion detection. Our results were very good and satisfying.

Keywords: smart detection system, deep learning, machine learning, object detection, server, and camera.

Résumé

Les régions désertiques de l'Algérie (le Sahara) sont connues (caractérisées) par la présence d'animaux venimeux de gravité variable (scorpions et serpents), qui choisissent des endroits difficiles pour se cacher, ce qui rend difficile de les voir et de les trouver. Par conséquent, le nombre élevé (pourcentage) de personnes empoisonnées, en particulier les enfants et les personnes âgées. Habituellement, on utilise des techniques traditionnelles pour détecter et attraper ces animaux venimeux, on les cherche et on les tue dès qu'on les voit. De plus, dans les zones où ces animaux vivent, on trouve des personnes qui élèvent des animaux qui ont la capacité de rechercher et de trouver ces animaux venimeux et de s'en débarrasser, comme les chats et les poules. Ce projet vise à détecter les serpents et les scorpions dans les zones résidentielles où ces animaux sont présents, ainsi qu'à fournir des avertissements opportuns et immédiats de leur présence. L'objectif est de prévenir et de protéger les habitants de ces zones contre les morsures venimeuses, qui peuvent entraîner des symptômes graves et, dans certains cas, même la mort.

Le travail présenté est un système intelligent de détection des serpents et des scorpions. Ce système se compose de deux appareils principaux, une caméra pour détecter les animaux venimeux et le téléphone du propriétaire pour envoyer et recevoir des alarmes et des notifications entre la caméra et le propriétaire. Le système est développé en utilisant des techniques d'apprentissage en profondeur et de vision par ordinateur. Nous avons utilisé le modèle YOLOv8 pour entraîner notre modèle de détection des serpents et des scorpions. Nos résultats étaient très bons et satisfaisants.

Mots-clés: système intelligent de détection, apprentissage en profondeur, apprentissage automatique, détection d'objets, serveur et caméra

Contents

List of Figures

Listings

List of Tables

General introduction

Snakes and scorpions are frightening creatures because of their toxic nature and its danger to human health. We find them in different part of the world, they have a special feature what make them very interesting especially for human safety.

Snakes with their smother bodies and venomous bite can cause neuromuscular paralysis (neurotoxicity), hemorrhage and coagulopathy (hemotoxicity), and/ local swelling, blistering, and tissue necrosis (cytotoxicity) around the bite site[42]. There are different kinds of venomous snakes and each poison has its own composition and effects. We can find that one poison is stronger than the other, but all kinds can threaten human's life especially those who live in an isolated area where you can't find medical care.

Scorpions also are not different from snakes, their venomous stings can be a serious threat to human life. Scorpion's sting causes a strong and intense pain, swelling and sometimes allergic reactions and other stings could be fatal because of the neurotoxins composition of their venom. In addition to being very painful, scorpion's stings can lead to complication particularly to those who have already health problems.

Snake and scorpion stings do not harm our bodies only but also it can cause psychological distress. Its impact creates a constant sense of anxiety, worry and insecurity.

Algeria is known for possessing the second largest desert in the world, which represents 80 of its total area. This desert is home to a diverse range of snakes and scorpions and is perfectly adapted to survive in harsh desert conditions. Snakes in the desert employ camouflage strategies with patterns and colors that blend with the desert sands, allowing them to remain hidden and difficult to spot or detect. Scorpions also always choose difficult places to hide what makes it hard for people of Sahara to see and find them. As a result the high number of poisoned people especially children and aged one.

In Algeria, 28 species and 14 genera of scorpions, classified under three families Buthidae, Chactidae (Euscorpiidae) and Scorpionidae were identified. The most important healththreatening scorpions found in Algeria include Androctonus australis and Leiurus quinquestriatus, and are found mostly in the southern highlands and in the Atlas and Hoggar mountain ranges. Other species also causing fatalities or threats to life are Androctonus aeneas, Buthus occitanus and Buthacus arenicola, found respectively in the southern highlands, in the northern Sahara, Hoggar and in the Tassili. All these species belong to the Buthidae family[25].

In the Algerian desert it is quite usual to come across snakes and scorpions due to the overlapping of their habitats with human settlement. People living in desert areas have gained a strong comprehension of these creatures, and employ various methods to reduce the risks caused by them, like growing a kinds of animals that have ability to search and find snakes and scorpions like cats and chickens. These traditional methods barely help to detect these creatures and avoid their stings. Yearly thousands of snakes and scorpions stings and tens of deaths are happening and the people of desert areas are living every day in stress and in fear of getting bite from those poisonous creatures.

In our project we aim to develop desert houses to smart houses by the installation of smart detection system that can identify, detect snakes and scorpions and also as a warning about it faster and as soon as possible to avoid and protect people of Sahara from the venomous stings and also providing psychological comfort and mind relaxing. Our system primary objectives can be summarized as follow:

- Detection of the venomous snakes and scorpions in desert houses.

- Warning and reporting of scorpions and snakes after the detection.

- Reducing the number of snakes and scorpions stings injuries.

- Providing protection and remove anxiety, worry and insecurity.

The structure of our work can be described as follow:

• Chapter 1: Generalities and contextualization this chapter starts with a presentation of smart home applications, the involved technologies, smart home advantages, generalities about machine learning and deep learning techniques.

• Chapter 2: Design and conception this chapter mention some of existing works (techniques used, accuracy, critics and limits), design of the system, global architecture and modeling using

UML (use case diagram, class diagram and sequence diagram).

• Chapter 3: Implementation and results this chapter illustrate the implementation of our system, presenting the used tools and technologies for the development of our system and provides evaluation results.

Finally, we conclude our Master report with a general conclusion and perspectives.

Chapter 1

Generalities and contextualization

1.1 Introduction

Smart home technology has transformed the way we interact with our living spaces, creating an intuitive, unprecedented convenience, efficiency, and control into our daily lives. With advancements in connectivity, automation, and artificial intelligence smart home technology has become more accessible and integrated than ever before.

Artificial intelligence (AI) is a wide-ranging field of computer science that focuses on creating intelligent computers that can accomplish jobs which would normally need human intellect. In this project we find machine learning (ML), deep learning (DL) and object detection which have demonstrated significant success in a variety of fields. As technology advances, machines are becoming increasingly capable of recognizing and detecting objects in images and videos with remarkable accuracy and speed.

In this this chapter, we introduce the general concepts of smart home technology. We start by giving definition of smart home, we present some of smart home applications and its benefits, the technology that smart home adopt. Following that an overview on Machine Learning, Deep Learning and object detection.

1.2 Smart home definition

A smart home means your home has a smart home system that connects with your appliances to automate specific tasks and is typically remotely controlled. You can use a smart home system

to program your sprinklers, set and monitor your home security system and cameras, or control appliances like your refrigerator or air conditioning and heating.[16] also smart home technology, often referred to as home automation, provides homeowners security, comfort, convenience and energy efficiency by allowing them to control smart devices, often by a smart home app on their smartphone or other networked device.[3]

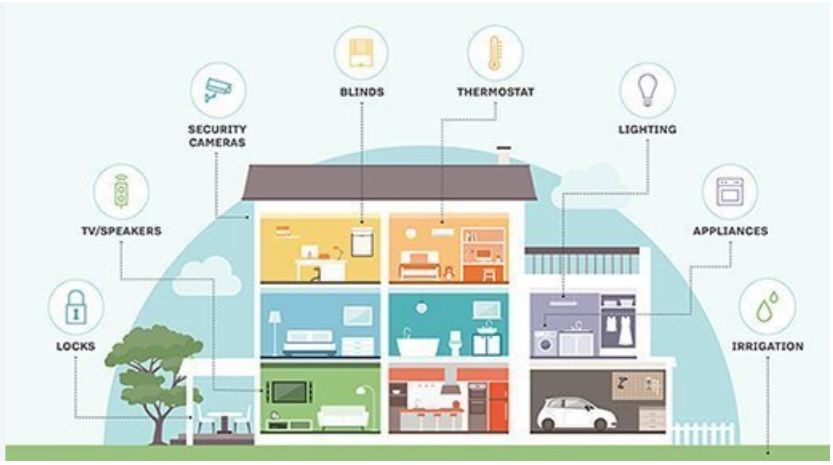


Figure 1.1: Smart home technology [44]

1.3 Smart home application

Smart home applications have their own requirements and challenges that should be considered when designing smart home protocols. We can classify smart home applications into different categories and extract the traffic characteristics and performance requirements for the different categories. Generally smart home applications can be classified into four main categories: security systems, energy management systems, health and safety systems and residents[38].

1.3.1 Security systems

A home security system is a group of physical electronic components that all work together to protect a home [41]. Smart home security systems are mainly used for monitoring the house and detecting if there is an unwanted intruder[38].

Often, a home security system will consist of the objects shown in the figure below:

Figure 1.2: Smart security systems [41]

1.3.2 Energy Management and Electrical Control Systems

Households use one of the major parts of the world's energy and more than half of the energy consumption in homes comes from electricity. The central task of energy management is to reduce costs for the provision of energy in households and residential building facilities without compromising the user's wellbeing[36]. Energy management is an important area of study for smart home applications. Several systems have been developed to reduce energy waste[38].

The functions of the home energy management are:

- Controlling activation/deactivation of home appliances[36].

- Collecting real-time energy consumption from smart meter and power consumption data from various household appliances.

- Generating and monitoring a dashboard to provide feedback about power usage.

1.3.3 Monitoring (Health/Elders/children) Systems

Having advanced technology in our homes will lead to various opportunities in the near future in this area. One of the most important is the monitoring of a person's cognitive and physical health and, as a consequence of an aging population, an area of critical need is eldercare.

The Health Smart Home concept, which can meet this challenge, has been extensively researched by many authors. One of the many benefits of monitoring human health by smart home [13]is the economic advantage of detecting diseases in their early stages, thus reducing overall healthcare costs.

Additionally, this technology offers patients, elders and those with disabilities the independence to live worry-free lives, as their medical data is continuously measured and sent to healthcare centers for prompt and efficient care. The work in [13] A system has been developed which utilizes sensors connected to a patient's body to collect data. This data is then transmitted through ZigBee technology to a healthcare center for analysis and monitoring.

1.3.4 Residents Convenience

The authors of [37]have created a fascinating home control and monitoring system, powered by an Android application reliant on Restful web services. The system consists of three components: a remote environment represented by a mobile application, a home gateway represented by a micro web server running on an Arduino with Ethernet technology, and a home environment containing appliances.

The architecture is not only cost-effective but also flexible and adaptable to different application scenarios. Additionally, any new devices added to the Arduino will be seamlessly integrated into the user's mobile application.

1.4 Benefits of smart home applications

We can summarize the benefits as follow: - Home monitoring with smart security system.

- Real-time alerts.

- Energy efficiency saving.

- Aid for elder residents.
- Chronic disease and disability management.
- Provide comfort and mental care health.

1.5 Smart home technologies

In most cases there are 4 technologies used in smart homes:

1.5.1 WiFi (based on IEEE 802.11)

Wi-Fi is a wireless networking technology that allows devices such as computers (laptops and desktops), mobile devices (smart phones and wearables), and other equipment (printers and video cameras) to interface with the Internet. It allows these devices–and many more–to exchange information with one another, creating a network[14].

1.5.2 ZigBee (based on IEEE 802.15.4)

Zigbee is a standards-based wireless technology developed to enable low-cost, low-power wireless machine-to-machine (M2M) and internet of things (IoT) networks[45].

1.5.3 Z-Wave

Z-Wave is a wireless communication protocol used primarily in smart home networks, allowing smart devices to connect and exchange control commands and data with each other. With two-way communication through mesh networking and message acknowledgment, the Z-Wave protocol helps alleviate power issues and brings low-cost wireless connectivity to home automation, offering a lower-power alternative to Wi-Fi and a longer-range alternative to Bluetooth[46].

1.5.4 Bluetooth

Bluetooth is a wireless technology that uses a radio frequency to share data over a short distance, eliminating the need for wires. You can use Bluetooth on your mobile device to share documents or to connect with other Bluetooth-enabled devices. For security reasons, Bluetooth devices

must be paired before they can begin transferring information. The process of pairing your devices will vary depending on the device you are connecting to[39].

1.6 Machine learning

Machine learning (ML) is a type of artificial intelligence (AI) (figure 1.3) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.[47] The main types of machine learning are (figure 1.4):

- Supervised learning.
- Unsupervised learning.
- Reinforcement learning.
- Semi- supervised learning.

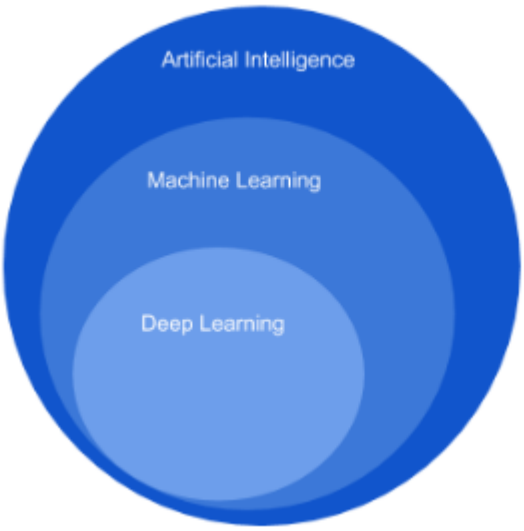


Figure 1.3: Relationship between AI, ML, and DL. [1]

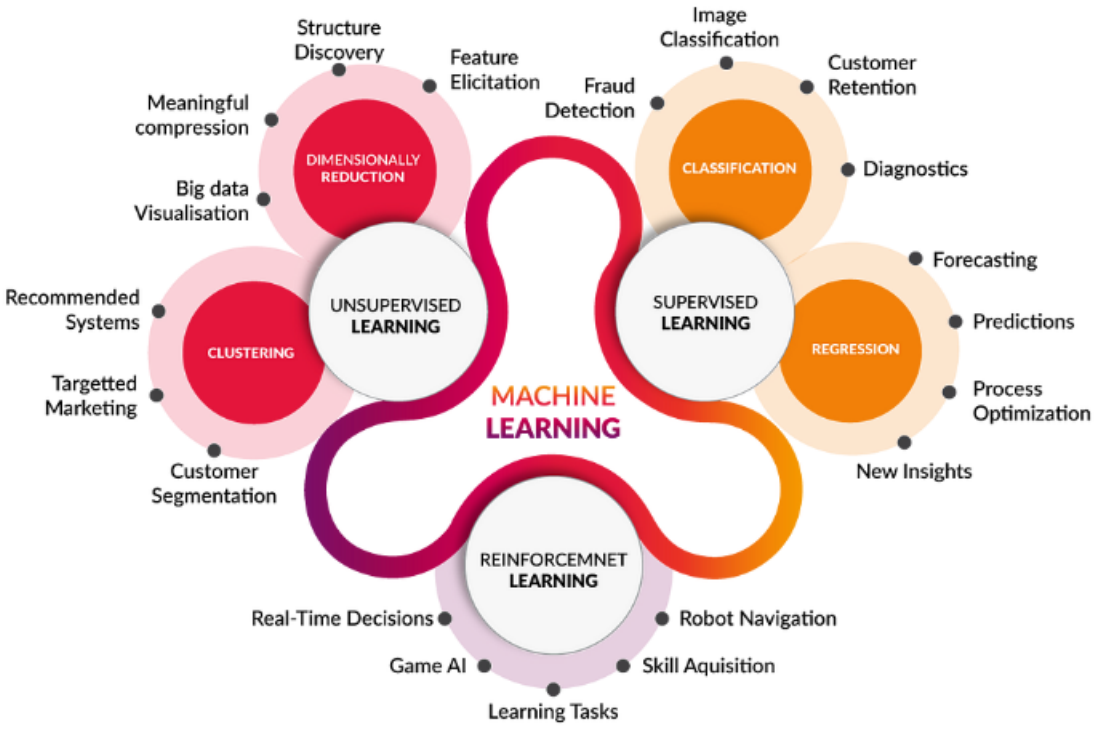


Figure 1.4: Machine learning and its types [43].

1.6.1 Supervised learning

Group of algorithms that require dataset which consists of example input-output pairs. Each pair consists of data sample used to make prediction and expected outcome called label. Word "supervised" comes from a fact that labels need to be assigned to data by the human supervisor. Supervised Learning models are trying to find parameter values that will allow them to perform well on historical data. Then they are used for making predictions on unknown data that was not a part of training dataset.

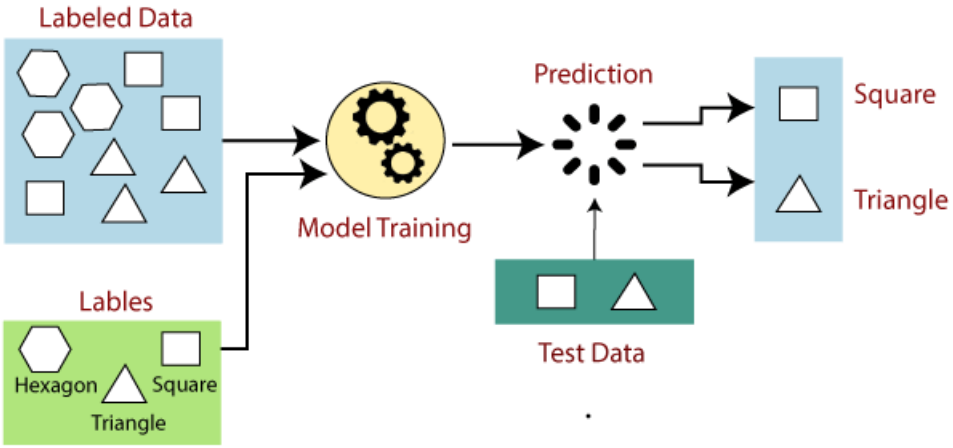


Figure 1.5: Example of supervised learning [28]

Supervised learning can be further divided into two types of problems:

1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends.[31] Some popular Regression algorithms which come under supervised learning are mentioned below:

- Linear Regression.
- Regression Trees.
- Non-Linear Regression.
- Bayesian Linear Regression.
- Polynomial Regression.

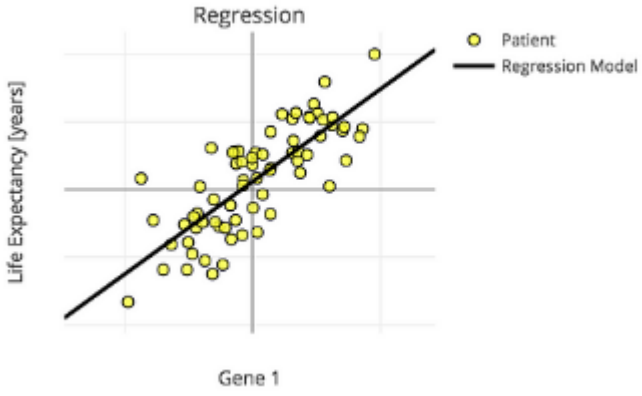


Figure 1.6: Example of regression model [21].

2. Classification

Supervised learning problem that involves predicting a class label [11]. In another definition, Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false. Some popular Regression algorithms which come under supervised learning are mentioned below:

- Logistic Regression.
- Decision Tree.
- Random Forest.
- Support Vector Machine (SVM).
- K-Nearest Neighbors (KNN).
- Naive Bayes

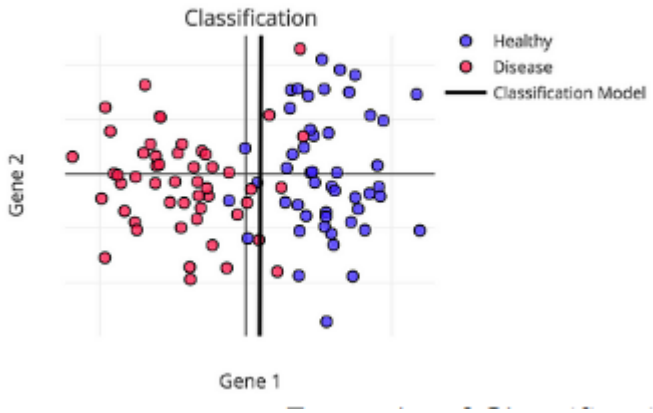


Figure 1.7: Example of classification model [22].

1.6.2 Unsupervised learning

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. Unsupervised learning describes a class of problems that involves using a model to describe or extract relationships in data [27].

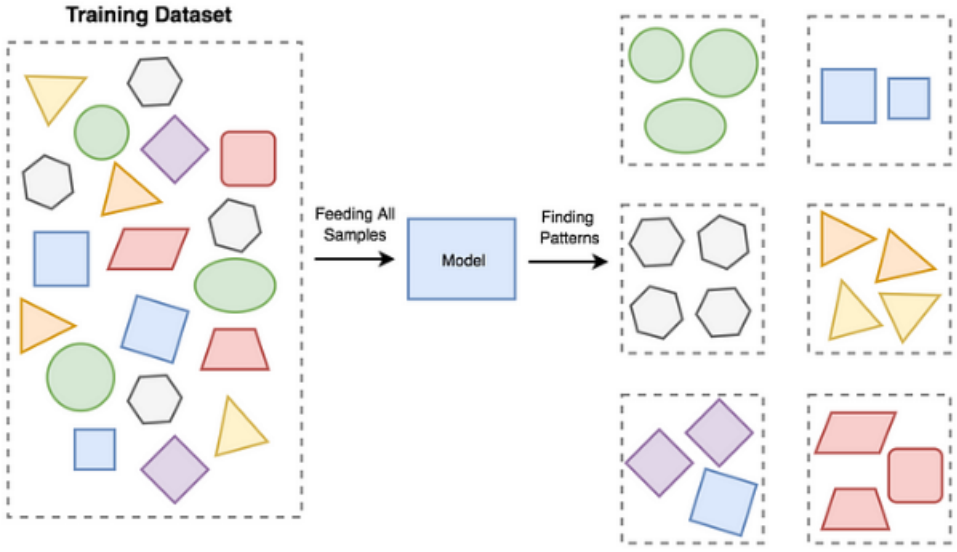


Figure 1.8: Example of unsupervised learning [20].

The unsupervised learning algorithm can be further categorized into two types of problems:

1. Clustering

Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group[27]. Clustering algorithms may be classified as listed below:

- Exclusive Clustering.
- Overlapping Clustering.
- Hierarchical Clustering.
- Probabilistic Clustering.
- K-means.
- Fuzzy K-means.
- Mixture of Gaussians.

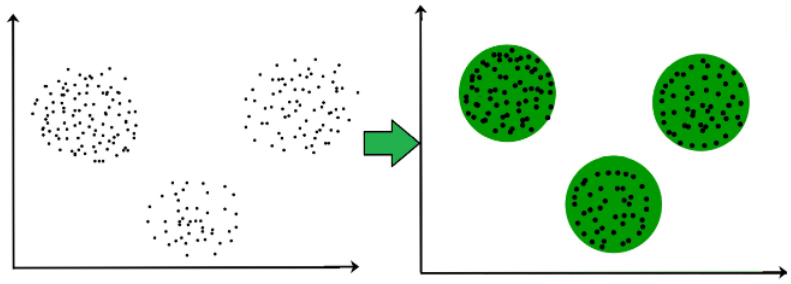


Figure 1.9: Example of clustering [23].

2. Association

An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset[30]. Association algorithms may be classified as listed below:

- Apriori Algorithm.
- Eclat Algorithm.
- F-P Growth Algorithm.

1.6.3 Reinforcement learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance [29].

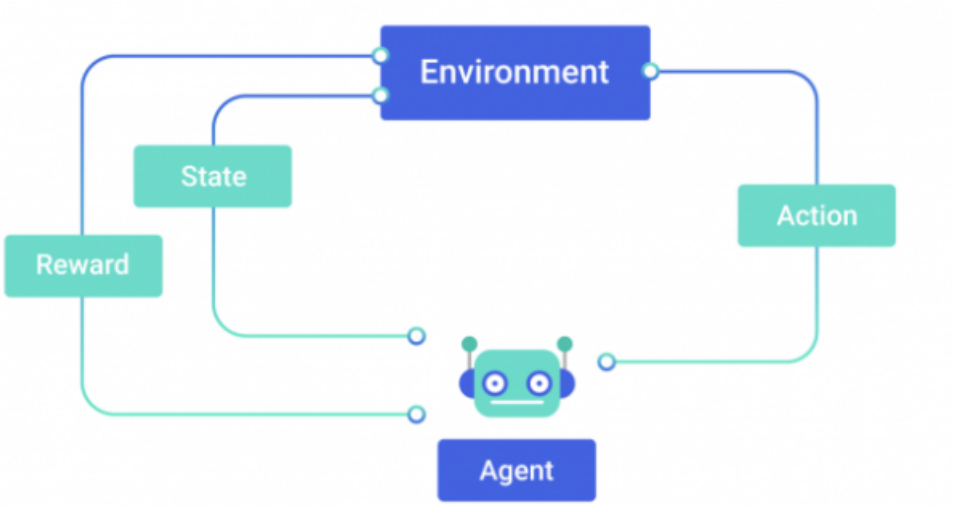


Figure 1.10: Example of reinforcement learning [48].

1.6.4 Semi-supervised learning

Semi-supervised learning is supervised learning where the training data contains very few labeled examples and a large number of unlabeled examples. The goal of a semi-supervised learning model is to make effective use of all of the available data, not just the labeled data like in supervised learning[11].

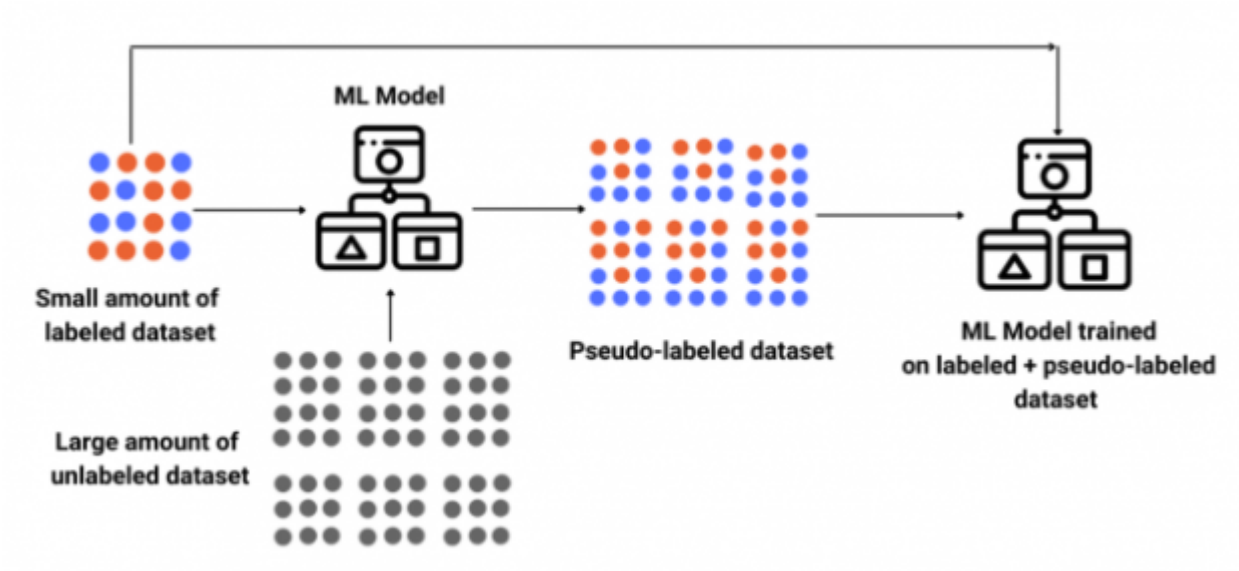


Figure 1.11: Example of semi-supervised learning [48].

1.7 Deep learning

1.7.1 Definition

Deep learning is a subset of machine learning (see figure 1.3), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention [26]. Deep learning is differentiated from traditional "shallow learning" because it learns much deeper levels of hierarchical abstraction and representations.

This learning technique is a groundbreaking tool for processing large quantities of data, since the performance of the machine improves as it analyzes more data. As the amount of data increases, the machine becomes more adept at recognizing even hidden patterns among the data. Because the machine is also learning from the processed data, it is able to perform feature extraction and abstraction automatically from the raw data with little to no human input[17].

1.7.2 Types of Deep Learning architectures

When it comes to deep learning, there are various types of neural networks. And deep learning architectures are based on these networks. We can indicate seven of the most common deep learning architectures[18]:

- Recurrent Neural Networks (RNN).
- Long Short-Term Memory (LSTM).
- Convolutional Neural Networks (CNN).
- Gated Recurrent Unit (GRU networks).
- Self-organized maps.
- Autoencoders.
- Restricted Boltzmann Machines

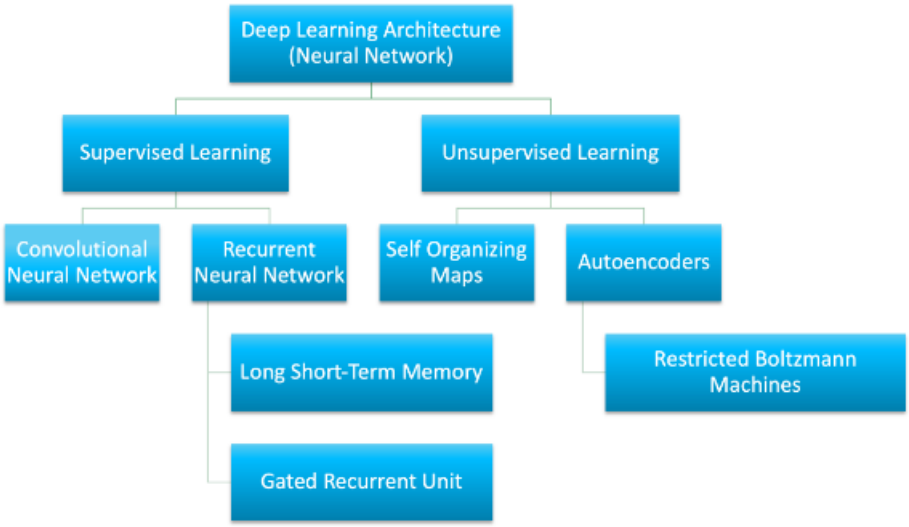


Figure 1.12: Types of deep learning architecture [18]

1.8 Artificial neural network (ANN)

ANNs are network of artificial neurons. Each ANN has a minimum of three layers, that is input layer that will take the input. Hidden layer that trains on the dataset fed to the input layer. The output layer that given output depending on input that is fed. artificial neural networks are a

subset of machine learning and are heart of deep learning algorithms. The layers in the neural networks are made of nodes (see figure 1.13).

A place where computation happens, loosely patterned on a neuron in the human brain, and when it encounters sufficient stimuli, it fires up, and it is called node. A node combines input that comes from the data, with weights that either can amplify or dampen the input, and so it will assign significance to inputs in regard to algorithm it's trying to learn [34].

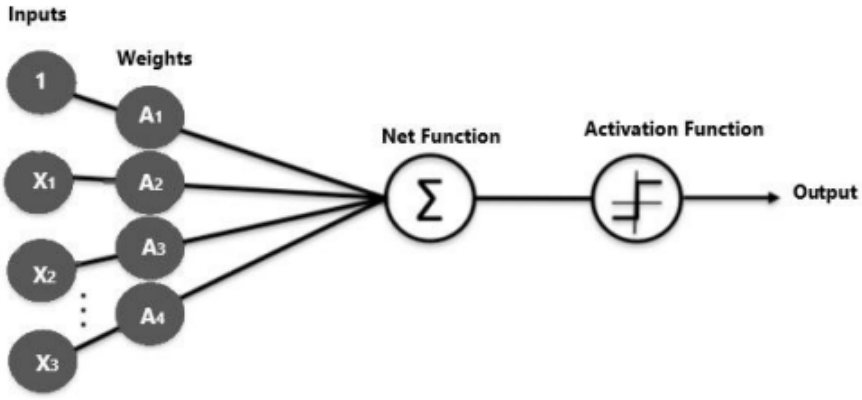


Figure 1.13: Node diagram [34].

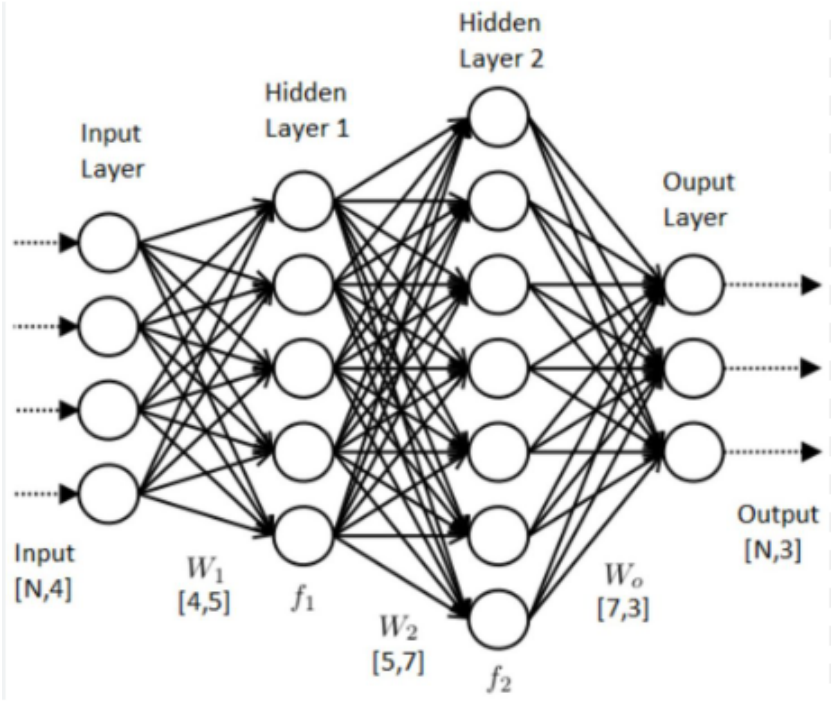


Figure 1.14: An artificial neural network[12].

1.9 Convolutional neural networks

1.9.1 Definition

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics[15].

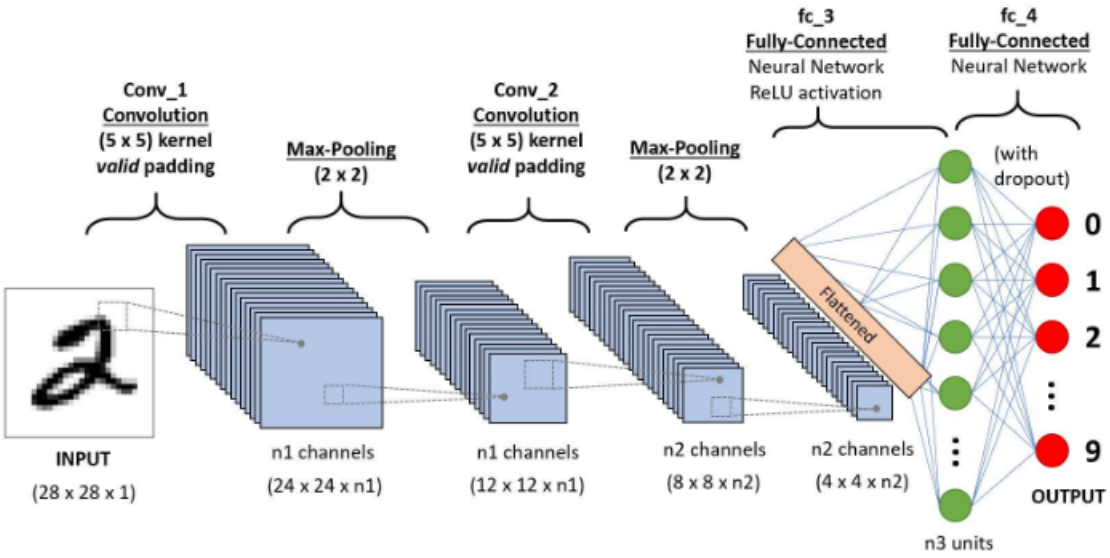


Figure 1.15: A CNN sequence to classify handwritten digits[15].

1.9.2 Convolutional Neural Network layers types

A CNN typically has five layers: a convolutional layer, a pooling layer, and a fully connected layer (figure 1.16):

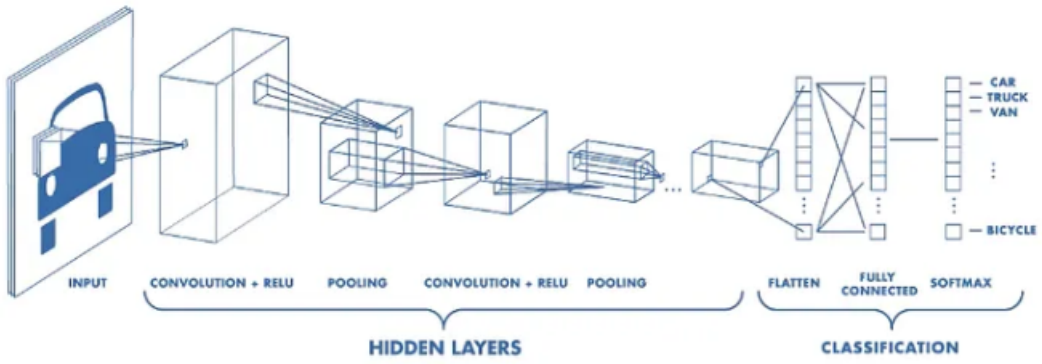


Figure 1.16: Architecture of a CNN[6].

1. Convolution Layer

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, (convolution operation and activation function). The convolution layer in CNN passes the result to the next layer once applying the convolution operation in the input[49]. Convolutional layers in CNN benefit a lot as they ensure the spatial relationship between the pixels is intact[9].

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map[9]. Depending upon method used, there are several types of Pooling operations. It basically summarises the features generated by a convolution layer. There two types of pooling layer:

- Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map[5].

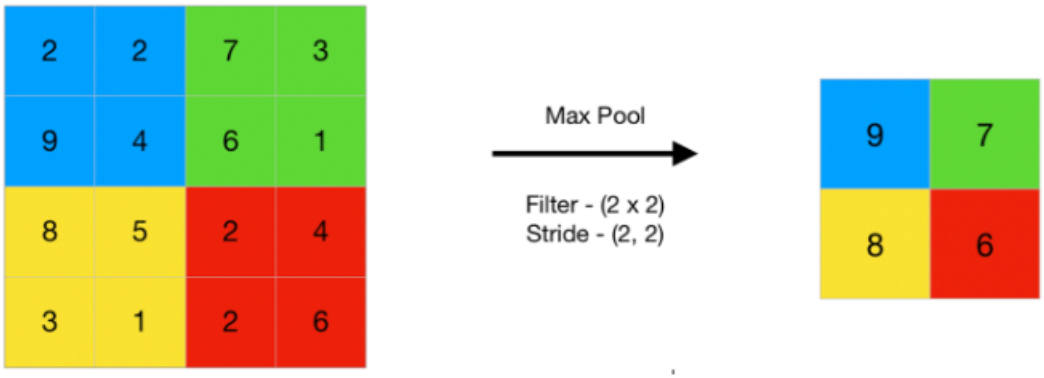


Figure 1.17: Max pooling[5].

- Average Pooling

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch[5].

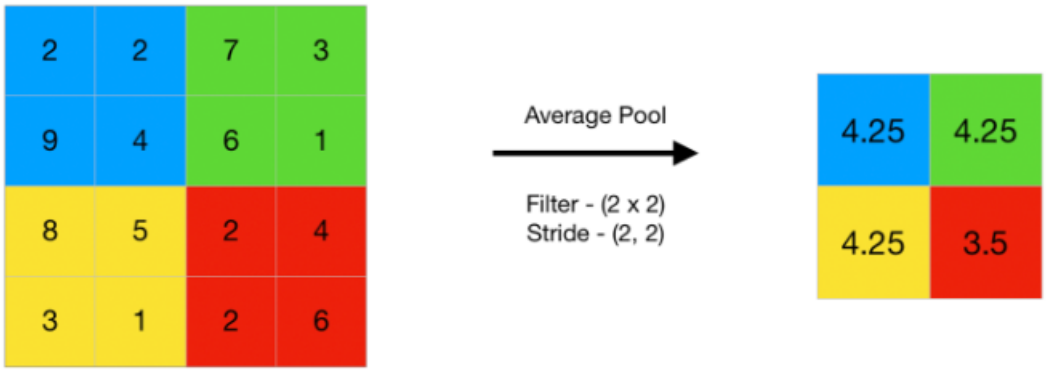


Figure 1.18: Average pooling[5].

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture[8].

4. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data

causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30 * of the nodes are dropped out randomly from the neural network. Dropout results[8] in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

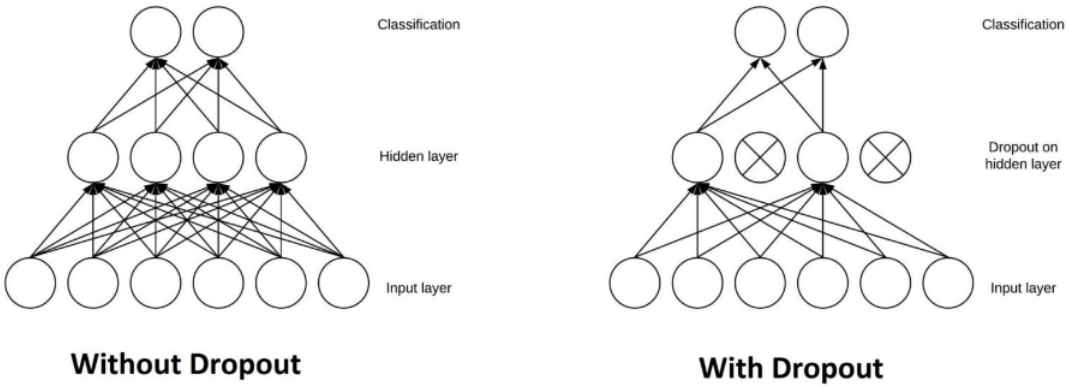


Figure 1.19: Example of dropout[4].

5. Activation Functions

One of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network.

There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred for a multi-class classification, generally softmax us used.

In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides whether the input to the work is important or not to predict using mathematical operations[7].

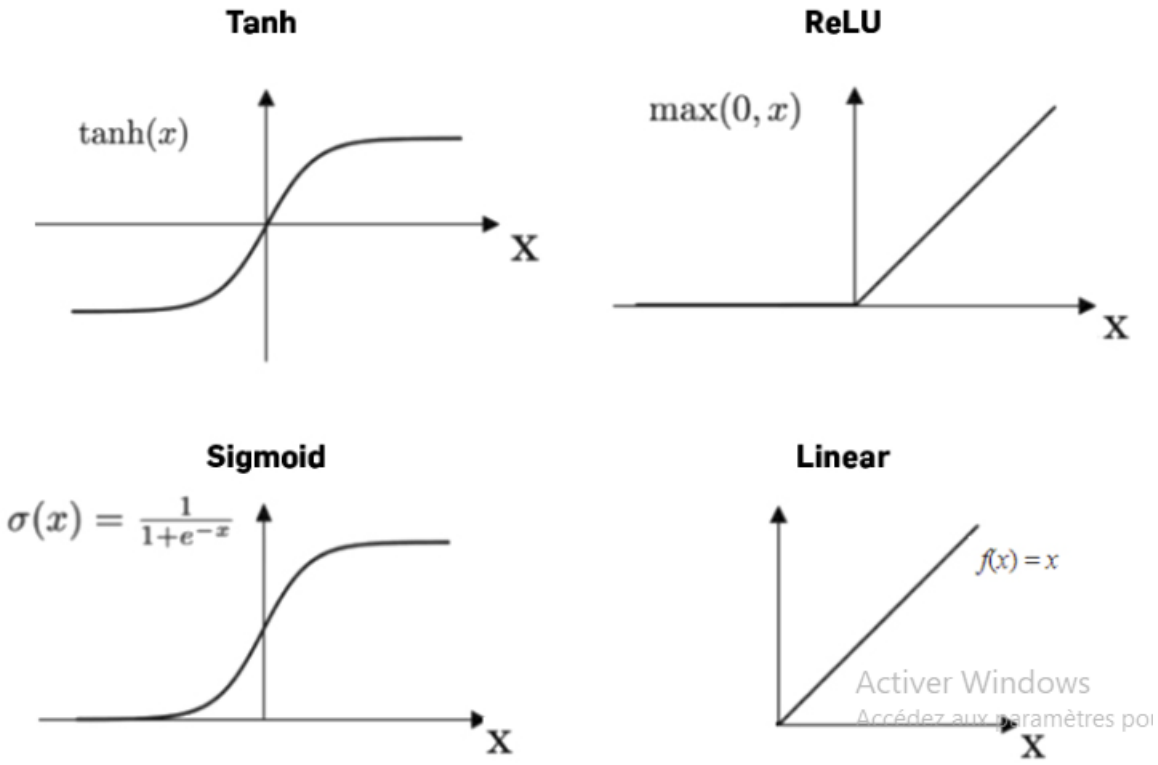


Figure 1.20: Activation functions[7].

1.9.3 CNN architectures

The number of CNN architectures are endless, we will go through some of the most popular ones:

1. LeNet

LeNet is the first CNN architecture. It was developed in 1998 by for handwritten digit recognition problems. LeNet was one of the first successful CNNs and is often considered the "Hello World" of deep learning. It is one of the earliest and most widely-used CNN architectures and has been successfully applied to tasks such as handwritten digit recognition. The LeNet architecture consists of multiple convolutional and pooling layers, followed by a fully-connected layer. The model has five convolution layers followed by two fully connected layers. LeNet was the beginning of CNNs in deep learning for computer vision problems. However, LeNet could not train well due to the vanishing gradients problem[32].

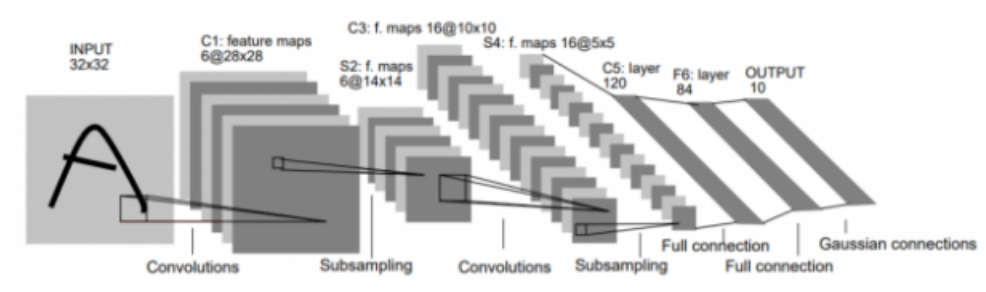


Figure 1.21: LaNet architecture[32].

2. AlexNet

AlexNet is the deep learning architecture that popularized CNN. AlexNet network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other. AlexNet was the first large-scale CNN and was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. The AlexNet architecture was designed to be used with large-scale image datasets and it achieved state-of-the-art results at the time of its publication. AlexNet is composed of 5 convolutional layers with a combination of max-pooling layers, 3 fully connected layers, and 2 dropout layers. The activation function used in all layers is Relu. The activation function used in the output layer is Softmax. The total number of parameters in this architecture is around 60 million[32].

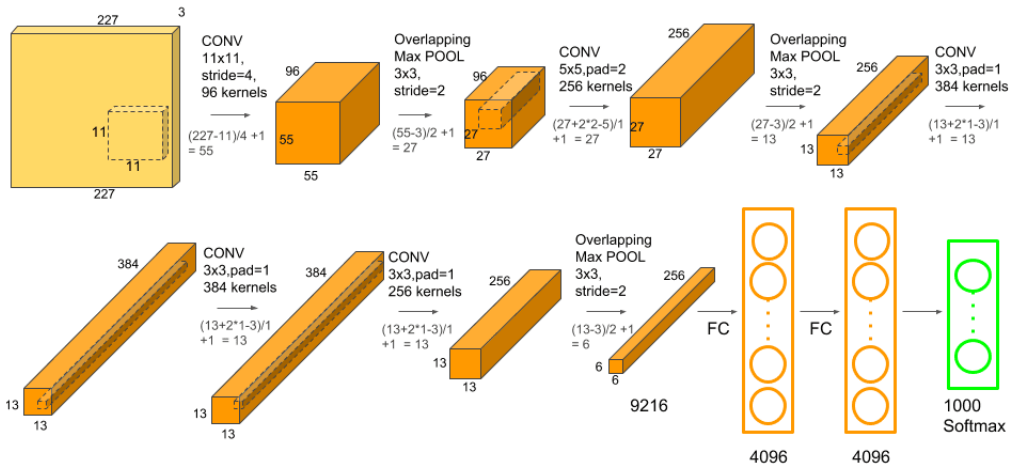


Figure 1.22: AlexNet – ImageNet classification[32].

3. GoogLeNet

GoogLeNet is the CNN architecture used by Google to win ILSVRC 2014 classification task. It has been shown to have a notably reduced error rate in comparison with previous winners AlexNet (Ilsvrc 2012 winner) and ZF-Net (Ilsvrc 2013 winner). In terms of error rate, the error is significantly lesser than VGG (2014 runner up). It achieves deeper architecture by employing a number of distinct techniques, including 1×1 convolution and global average pooling. GoogleNet CNN architecture is computationally expensive[32].

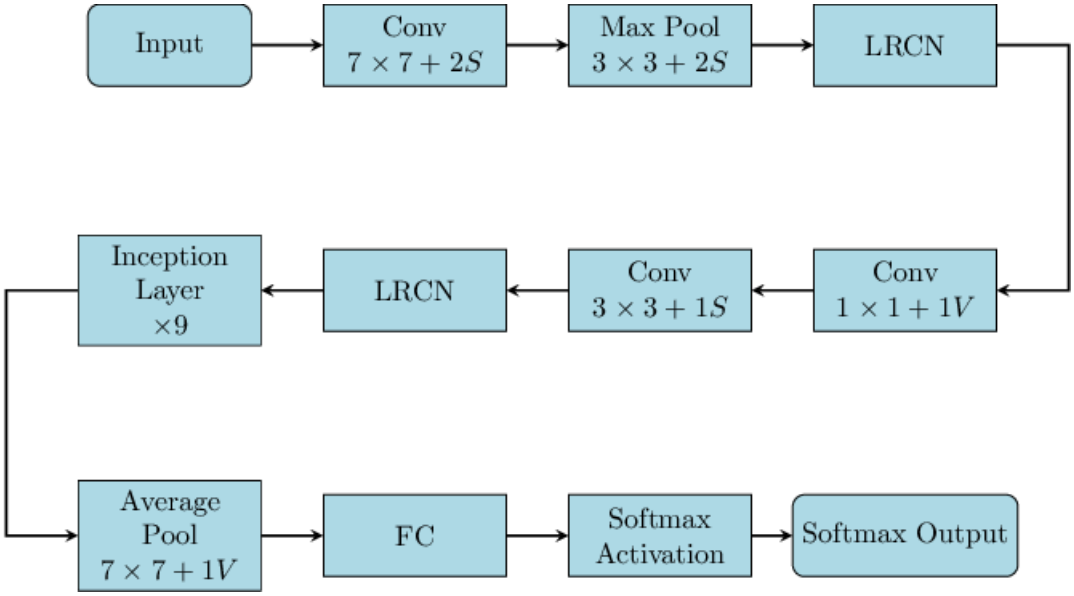


Figure 1.23: A simplified block diagram of the GoogLeNet Architecture[32].

4. VGGNet

VGGNet is the CNN architecture it is a 16-layer CNN with up to 95 million parameters and trained on over one billion images (1000 classes). It can take large input images of 224 x 224-pixel size for which it has 4096 convolutional features. CNNs with such large filters are expensive to train and require a lot of data, which is the main reason why CNN architectures like GoogLeNet (AlexNet architecture) work better than VGGNet for most image classification tasks where input images have a size between 100 x 100-pixel and 350 x 350 pixels[32].

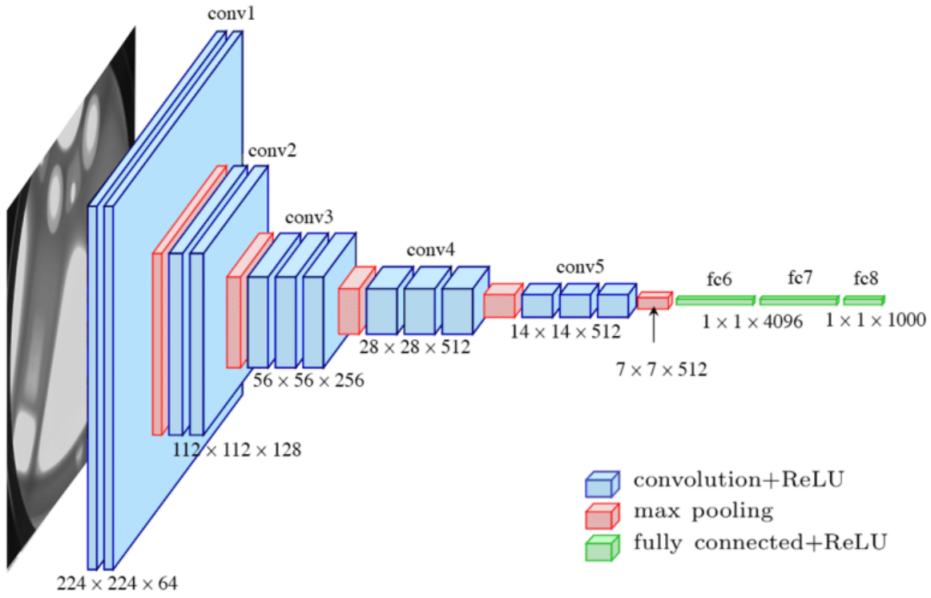


Figure 1.24: The standard VGG16 network architecture diagram[32].

1.10 Transfer learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems[10].

1.11 Object detection

1.11.1 Definition

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this

intelligence using a computer[35].

1.11.2 Difference between object detection and image recognition

Image recognition assigns a label to an image. A picture of a dog receives the label “dog”. A picture of two dogs, still receives the label “dog”. Object detection, on the other hand, draws a box around each dog and labels the box “dog”. The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition[2].

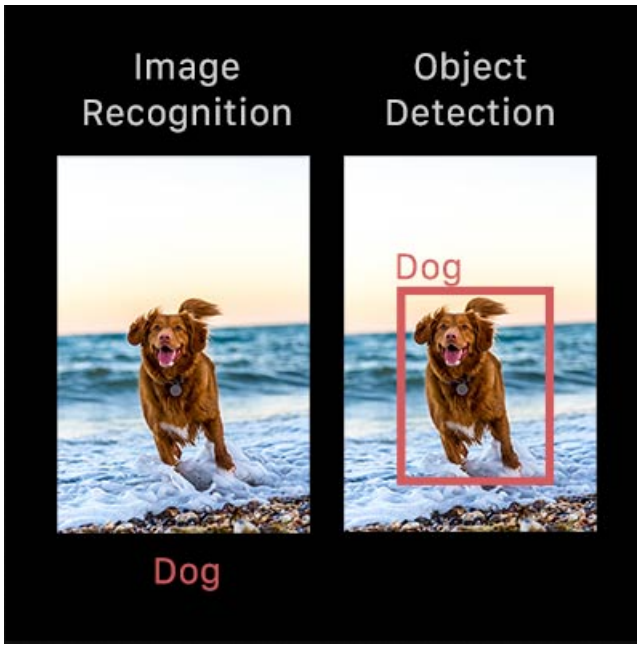


Figure 1.25: The difference between image recognition and object detection[2].

1.11.3 Object detection algorithms

Object detection algorithms are generally classified into two categories, One-stage detector and Two-stage detector as it shows in the figure 1.26 below:

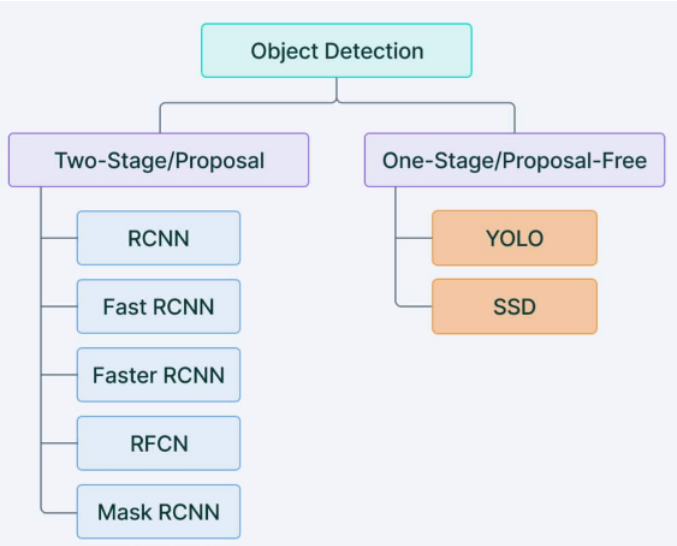


Figure 1.26: object detection algorithms [33]

1.11.4 YOLO (you only look once)

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images. YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously[40].

YOLO has gone through several iterations and improvements and the figure 1.27 below shows YOLO models timeline:

Figure 1.27: Release date timeline of YOLO models[19].

1.12 Conclusion

In this chapter, we covered some of the most important aspects of our current work We discussed smart home technology, smart home application, benefits of the applications and the involved technology for a smart home then We detailed the basics of machine learning and deep learning and finally we talked about object detection, the difference between it and image recognition and the main object detection algorithms.

Chapter 2

Design and conception

2.1 Introduction

During the last few decades machine learning, deep learning and object detection have advanced significantly and witnessed remarkable progress, transforming numerous industries and benefiting people in various ways. Since researches progress and technology development, the impact of machine learning, deep learning and object detection grow; these technologies hold immense potential in addressing complex challenges and enhancing human capabilities across industries. In this chapter, we have presented the general idea of the project, we have mentioned few related works, and also we have presented the system global architecture, then we have modeled our system using UML [unified modeling language] by creating the necessary diagrams that can help us understand and analyze our system.

2.2 General idea of the project

To ensure safety from the venomous stings of snakes and scorpions for people who lives in desert areas we have developed a smart detection system designed particularly in snakes and scorpions detection. This system consists of a camera for the detection and mobile application in which the users get notified and receive warning message about these creatures, the warning message comes in form of an image of the snake or scorpion which has been detected with its location (figure 2.1).

We have used one of the deep learning techniques which is represented in CNN more spe-

cific YOLO algorithm for object detection. Scorpion and snake detection with YOLO uses Convolutional Neural Networks (CNNs) for object detection, specifically for predicting the bounding boxes and class probabilities for scorpions and snakes in an image. Supervised learning is used to train the YOLO model on a dataset of scorpion and snake images labeled with bounding boxes indicating the scorpion's and snake's location.

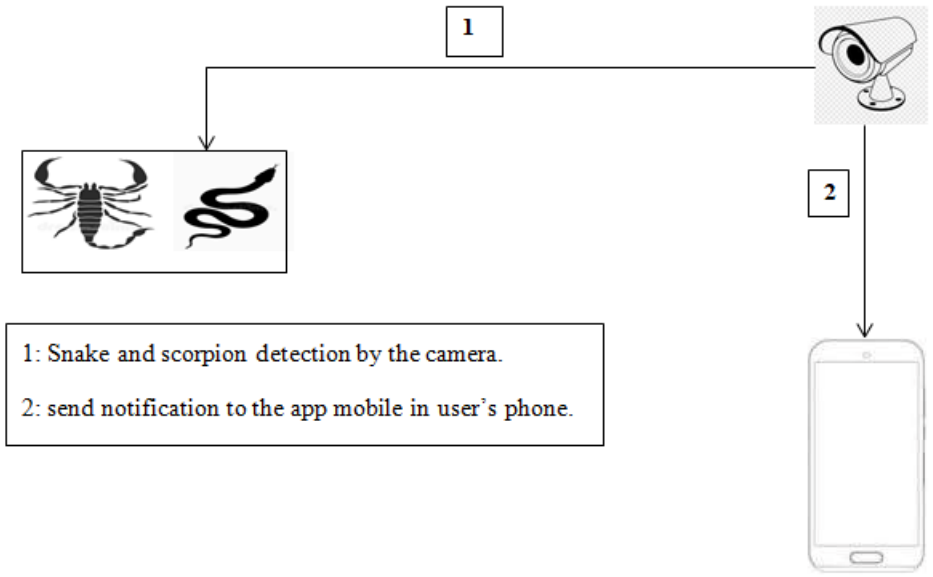


Figure 2.1: General idea design.

2.3 Related work

We will mention some of the previous works on detecting snakes and scorpions using DL and ML techniques in the table 2.1:

Reference	Topic	Description	Limits
[24]	Scorpion detection and classification systems based on computer vision and deep learning for health security purposes	This article focuses in the development of scorpion detection and classification systems based on computer vision and deep learning for health security purposes. The researchers used two different deep learning models, YOLOv4 and MobileNetV2, for object detection and image classification. Various metrics such as accuracy, precision, recall, and F-measure are used to evaluate these models. The results show that both YOLOv4 and MobileNetV2 models achieved high accuracy values of 88 * and 91 * , respectively, for scorpion detection. However, the MobileNetV2 model showed better performance than YOLOv4 in terms of recall value (97 vs. 90), indicating that it was better at correctly identifying true positives. This article also demonstrated the potential applications of deep learning in health security related to scorpions, such as early warning systems and public health interventions.	We have noted that The study was conducted under controlled laboratory conditions using specific datasets thus, the performance of the developed systems may be different in real-world scenarios and with different datasets. The focus of the study was to distinguish between two genera of scorpions (Tityus and Bothriurus), and it is unclear how well the developed systems would perform with other genera and species of scorpions

[50]	Snake Detection and Classification using Deep Learning	This article focuses on the detection and classification of snakes using deep learning models in complex images. It highlights the importance of snake detection in a variety of applications, including wildlife conservation and human safety. different deep learning models are explored by this article, including YOLOv3, Tiny YOLO, SSD MobileNet, SSD VGG16, Faster RCNN ResNet, and RetinaNet These models are trained on a dataset of snake images to learn the features necessary for accurate detection and classification. The results demonstrate that Faster RCNN ResNet outperforms Tiny YOLO in terms of accuracy, with a mAP of 81.62 versus 66.23 for Tiny YOLO. The article contains an experiment for detecting and classifying a single snake in a picture. The Faster RCNN ResNet model successfully detects almost all snake instances, achieving nearly 100 recall. In this experiment, the embedded Tiny YOLO model performs relatively poorly.	a limited data set is used in this study and that may cause an overfitting ,also while this article discusses the performance of different models, it does not provide detailed insights into the limitations or weaknesses of each model. It primarily focuses on the accuracy and inference time of the Tiny YOLO and Faster RCNN ResNet models. The article mentions the training and evaluation hardware, such as GPU and RAM specifications. It does not, however, describe any limitations or potential influence of hardware constraints on the models' performance or scalability.

Table 2.1: Related works description and limits.

2.4 Global architecture of the system

The figure 2.2 shows the global architecture of our system:

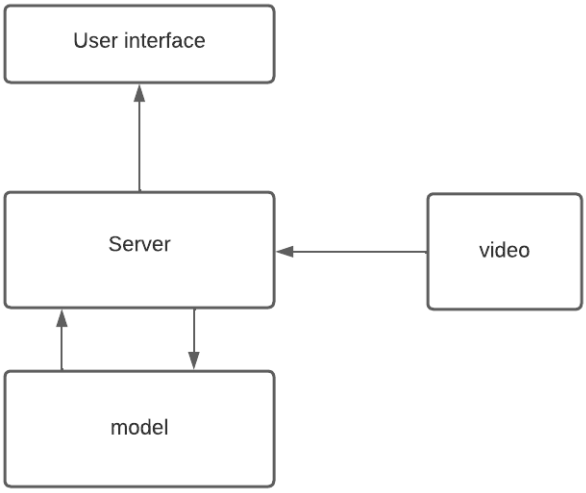


Figure 2.2: Diagram of the global architecture of the system.

A video is captured by the camera and send to the server which will divide the video into several frames and send each frame to the model, the model will apply the detection process on all the frames, if any snake or scorpion is detected the server will send an alarm, a sms message and email to the user mobile phone (user interface) .

The process used to detect snakes and scorpions in images using DL begins by collecting the dataset and applying the needed preprocess to improve and enhance the images. Then, it is fed to the DL model.

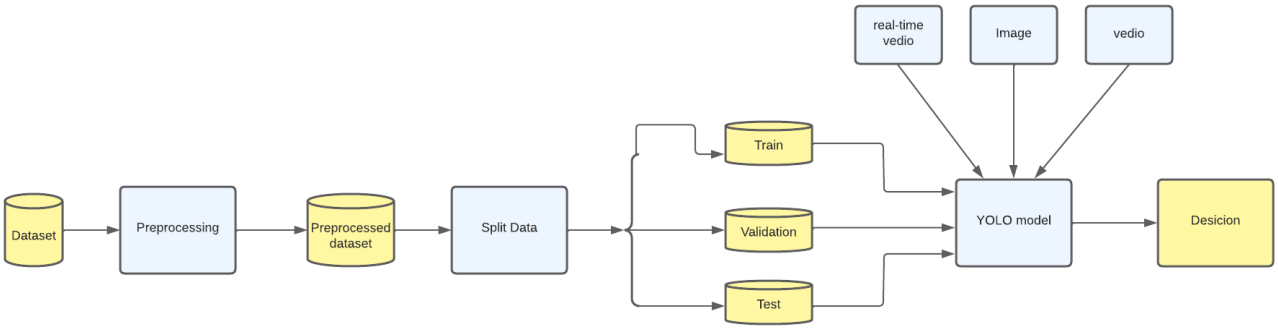


Figure 2.3: The process of detecting snakes and scorpions.

2.5 System conception

This section provides the modeling of our system using UML to elaborate the use case diagram, class diagram and sequence diagrams.

2.5.1 UML definition

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

2.5.2 Class diagram

The figure 2.4 presents the class diagram of our system:

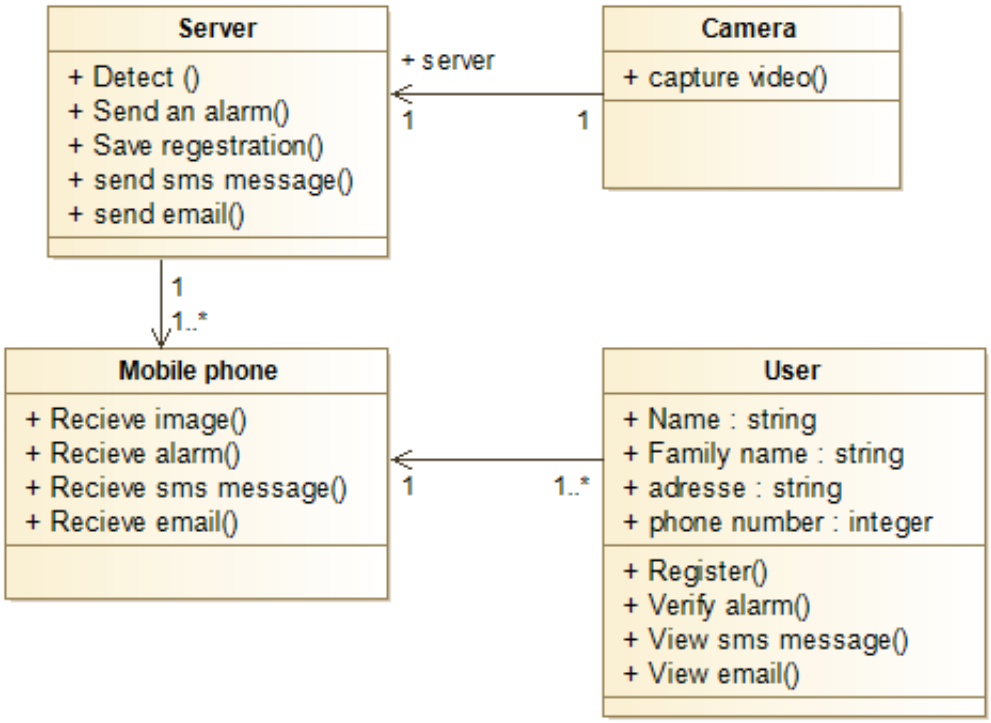


Figure 2.4: Class diagram of the system.

The table below (Table 2.2) explains each class in the class diagram shown above (figure 2.4):

Class	Explanation
Camera	It contains only one method which is capture video, each camera has a server.
Server	It contains five methods represented in: save registration, detect (the detection process) and send alarm to the mobile phone,. Contains 1 camera, and from 1to n number of mobile phone.
Mobile phone	It contains four methods: receive images, receive an alert, receive a sms message and receive an email
User	It contains four attributes: name, family name, phone number and address, also contains four methods: register, verify alarm, view sms messages and view email.

Table 2.2: Class diagram explanation.

2.5.3 Use case diagram

We can elaborate 2 use case diagrams for the system:

Use case diagram for camera

The figure 2.5 presents the use case diagram of the camera:

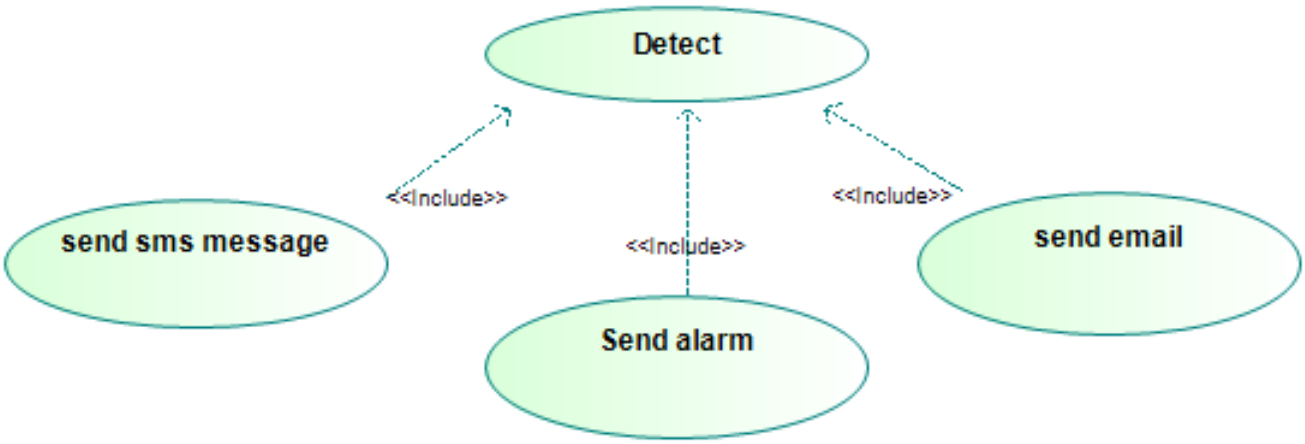


Figure 2.5: Use case diagram of the camera

The table below (Table 2.3) presents a description of the use case diagram:

Use case	Description
Detect	Detect snakes and scorpions in the house using a camera.
Send alarm	Camera sends an alert message (alarm) to the mobile phone when a snake or scorpion is detected.
Send sms message	Adding to the alarm the camera sends sms messages to homeowner mobile phone in order to warn them.
Send email	An email also is send to the mobile phone of the home owner the email contain the image of the snake or the scorpion detected.

Table 2.3: Description of the use case diagram for camera.

Use case diagram for mobile application

The figure 2.6 presents the use case diagram of the mobile application:

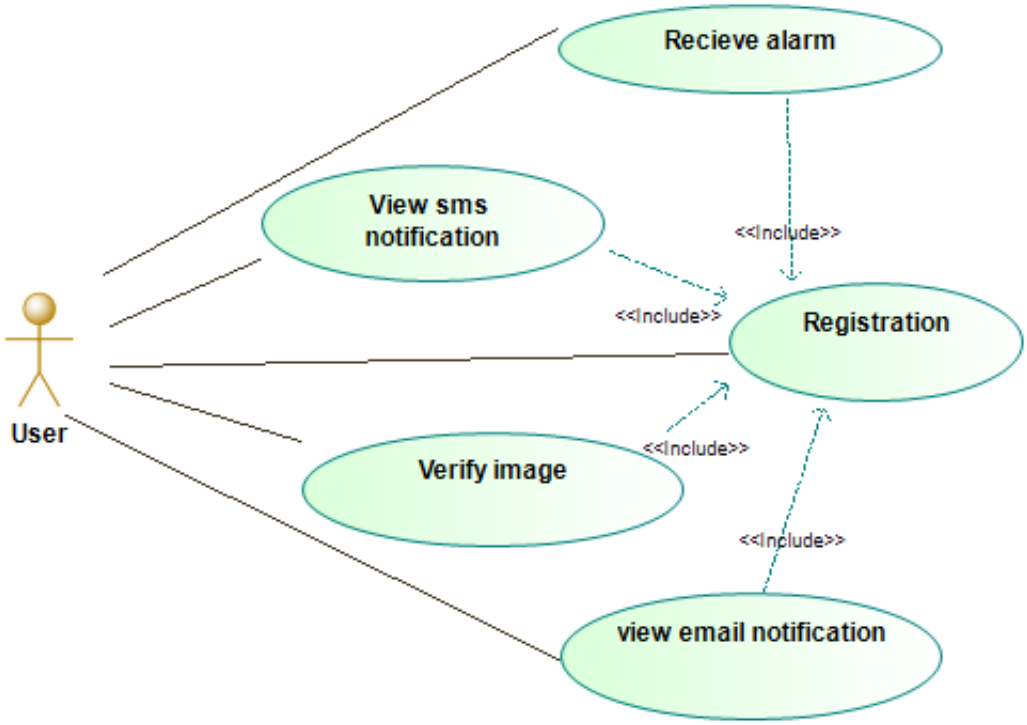


Figure 2.6: Use case diagram of the mobile phone.

The table 2.4 below presents a description of the use case diagram:

Actor	Use case	Description
User	- Registration - Verify image - View sms notification - View email notification - Receive alarm	Users must register to create an account. The phone receives a notification (alert) from the camera that a snake or scorpion has been detected. Homeowner can verify and view notification from his mobile. Homeowner can view his history and also can see the amount of detecting snakes and scorpions.

Table 2.4: Use case diagram from mobile application description.

2.5.4 Sequence diagram

This section is concerned about modeling the processes of our system using the sequence diagram (figure 2.7):

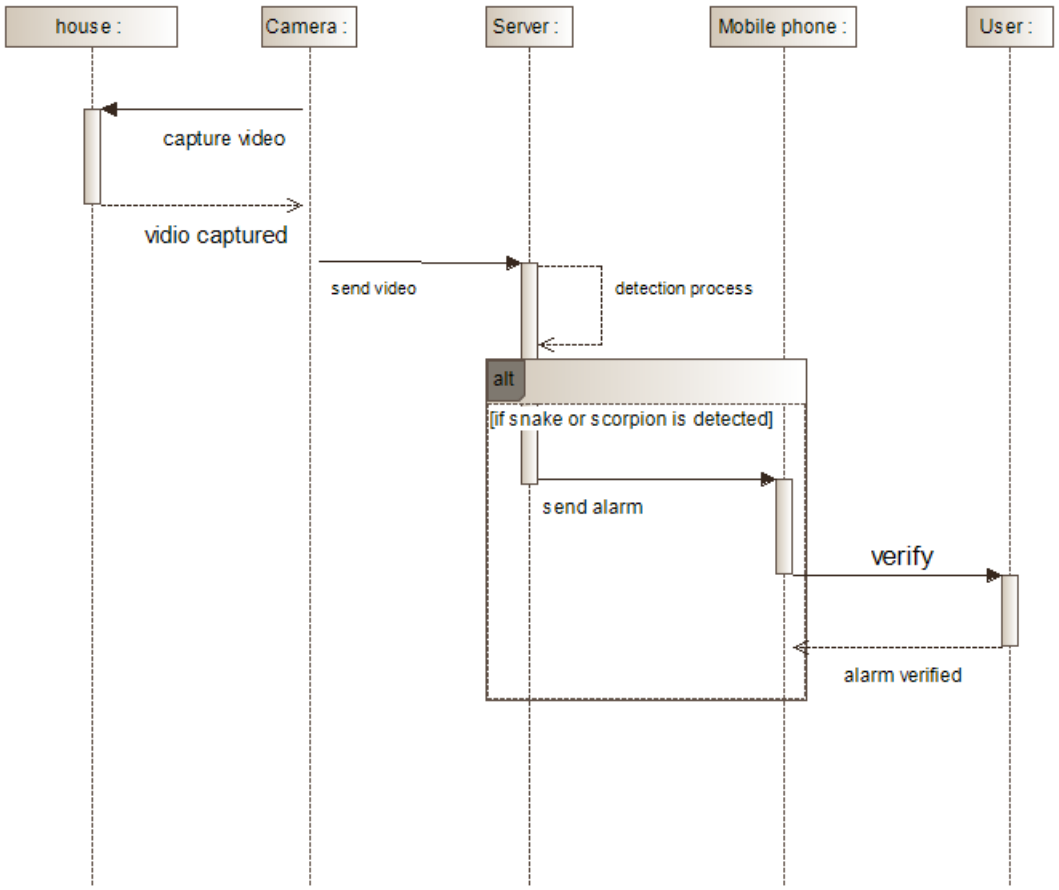


Figure 2.7: Sequence diagram of the system.

2.6 Conclusion

The purpose of this chapter was to present the general idea of the project, describe our system design, global architecture and the process of detecting snakes and scorpions in image. And finally we modeled our system using UML (class diagram, use case diagram and sequence diagram).

The next chapter (chapter 3), we will present the implementation of our system, the used tools and discuss the results.

Chapter 3

Implementation and results

3.1 Introduction

In the previous chapter, we presented a full explanation of our system's general architecture as well as covered the design processes of the snakes and scorpions detection and used the UML to model our system.

We begin this chapter by discussing the platforms and tools that were employed in the development of our system. We have presented the implementation of all parts of the system and we will present and discuss the obtained results.

3.2 Development tools, frameworks and libraries

In this section, we will cover the tools and platforms that we utilized in our system:

• Google Colab

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

Figure 3.1: Google colab logo.

• **CVAT**

CVAT stands for Computer Vision Annotation Tool; it is a free, open-source digital image animation tool written in Python and JavaScript. CVAT supports supervised machine learning tasks for object detection, image classification, image segmentation, and 3D data annotation.

Figure 3.2: CVAT logo.

• **OpenCV**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Figure 3.3: OpenCV logo

• **Roboflow**

Roboflow is a computer vision platform that allows users to build computer vision models faster and more accurately through the provision of better data collection, preprocessing, and model training techniques. Roboflow allows users to upload custom datasets, draw annotations, modify image orientations, resize images, modify image contrast and perform data augmentation. It can also be used to train models.

Figure 3.4: Roboflow

• **Anaconda spyder**

Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection and beautiful visualization capabilities of a scientific package.

• **Matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

• **Keras**

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

• **TensorFlow**

TensorFlow is an open source framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers.

Figure 3.5: Tensorflow logo.

• **NumPy**

NumPy is an open source project that enables numerical computing with Python. It was created in 2005 building on the early work of the Numeric and Numarray libraries.

• **Twilio python API**

The Twilio python API offers a vast set of services including SMS, MMS, programmable voice messages, WhatsApp API integration and many more. One of the highlighted features among them all is sending and receiving SMS and MMS messages.

Figure 3.6: Twilio logo.

• Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Figure 3.7: Python logo.

• Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs

- light on a sensor, a finger on a button, or a Twitter message

- and turn it into an output

- activating a motor, turning on an LED, publishing something online.

• ESP32-CAM

The ESP32-CAM is a small size, low power consumption camera module based on ESP32. It comes with an OV2640 camera and provides onboard TF card slot. The ESP32-CAM can be

widely used in intelligent IoT applications such as wireless video monitoring, WiFi image upload, QR identification, and so on.

Figure 3.8: ESP32-CAM.

3.3 System Implementation

In this section of the chapter, we provide a full overview of our system, accompanied by screenshots:

3.3.1 ESP32-camera implementation

We have presented the implementation of ESP32 camera in details: First step must be done to start the implementation of ESP32 camera is to install the driver of ESP32 on the arduino platform.

To launch the camera we need to change few things in the example sketch where we can find an example code for the ESP32 camera steps mentioned bellow describes how to get to the code:

- Launch the Arduino IDE and navigate to File > examples
- Choose ESP32 > Camera > CameraWebServer.

In the code we must mention the name of the Wi-Fi network and the password (figure 3.9) taking into consideration that all the system (camera, server and phone) must connect to the same Wi-Fi network.

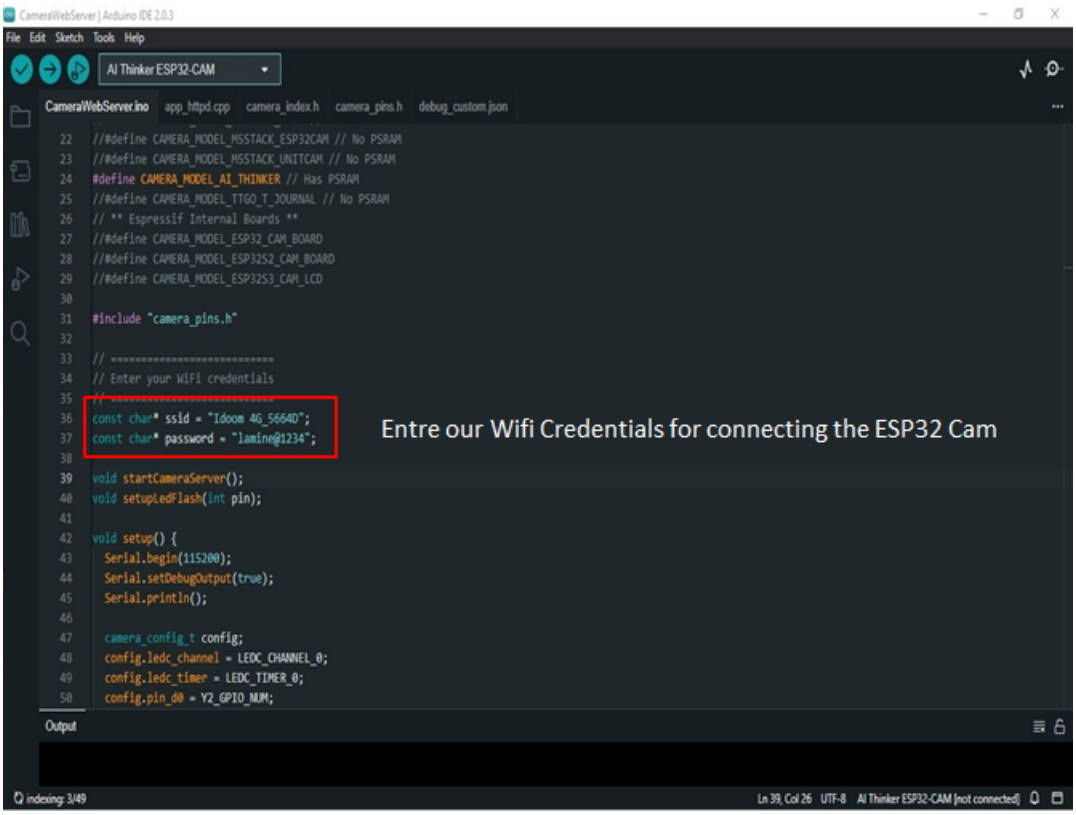


Figure 3.9: Connecting the ESP32 camera to the Wi-Fi network

ESP32 CAM doesn't have an onboard programming chip, so we have to add an external USB that connect between the camera and the android platform(figure 3.10) after that we need to select board and the port to allow the uploading process to start.

The steps we need to select a board and port:

- Go to Tools and select the right board which is in our case AI-Thinker ESP32-CAM
- Select the right port in our case "COM5" and hit upload

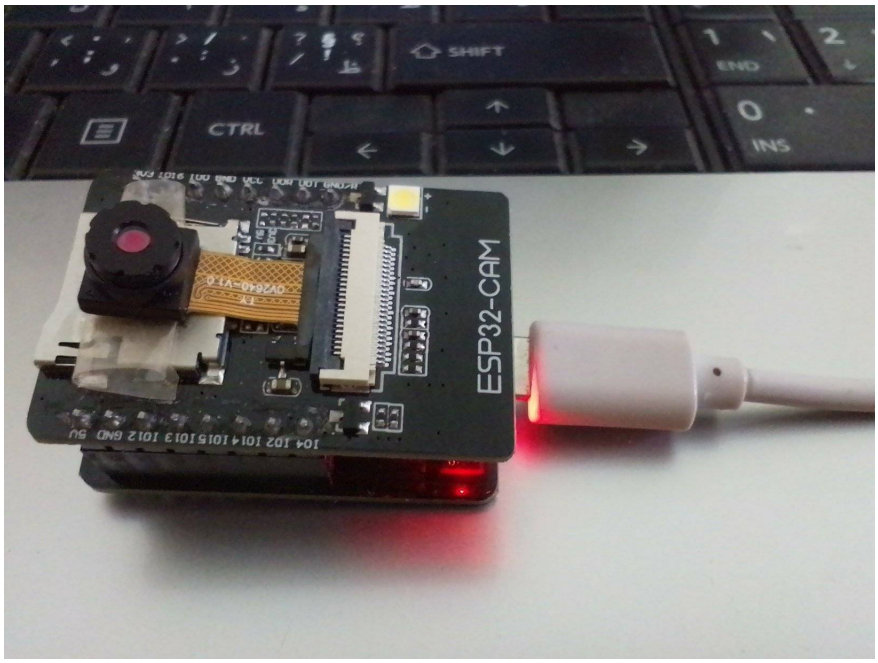


Figure 3.10: Connect the ESP32-cam with USB

After we save and compile, the code will be uploaded into our ESP32 camera as shown in the output of figure 3.11 bellow:

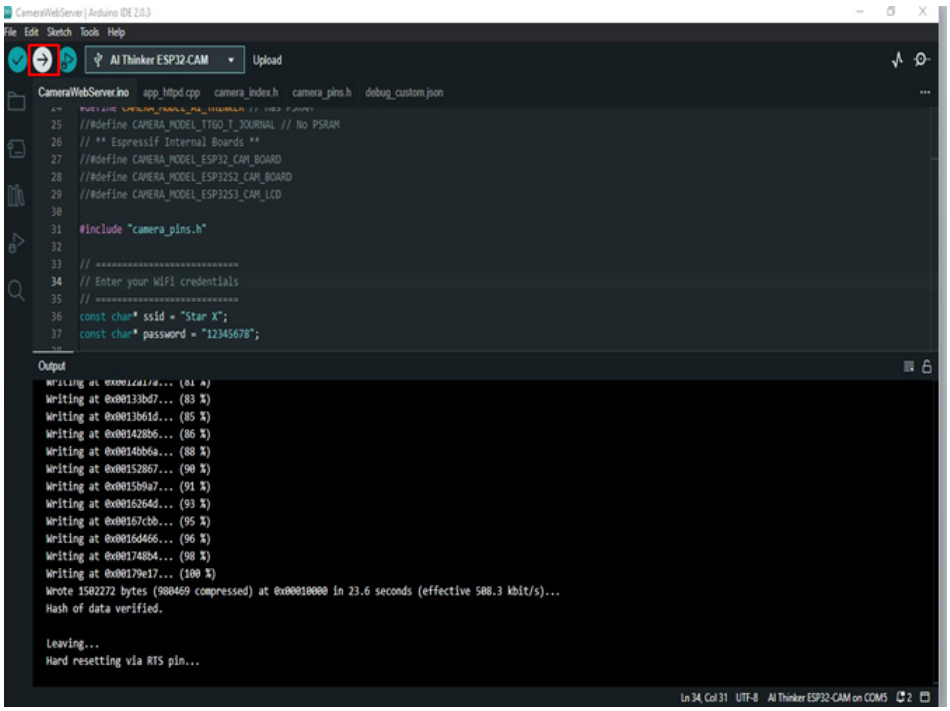


Figure 3.11: Upload the code source in the ESP32 camera.

After uploading our code in ESP32 cam, we must connect our camera to a browse in order to control it. For that we need to get the link of the camera from the Serial Monitor and past it the browser then we can turn our camera on by clicking the start stream button. The steps are shown in figure 3.12 and figure 3.13 bellow:

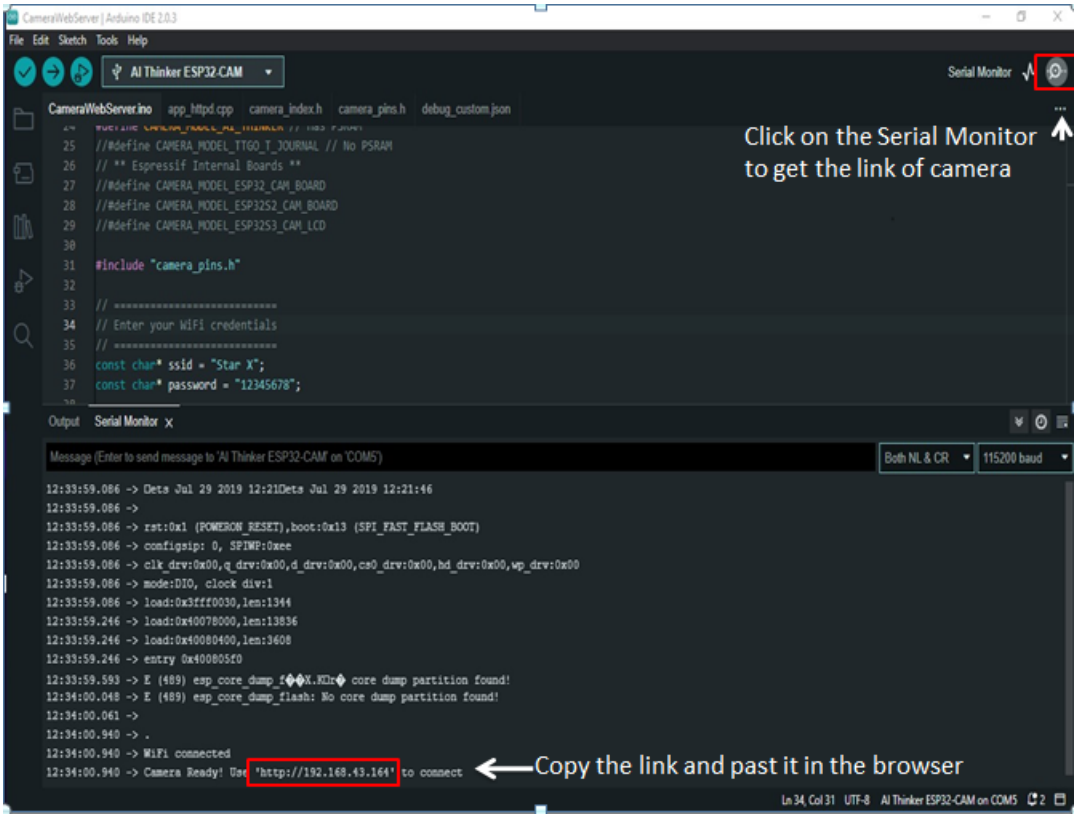


Figure 3.12: get camera's link.

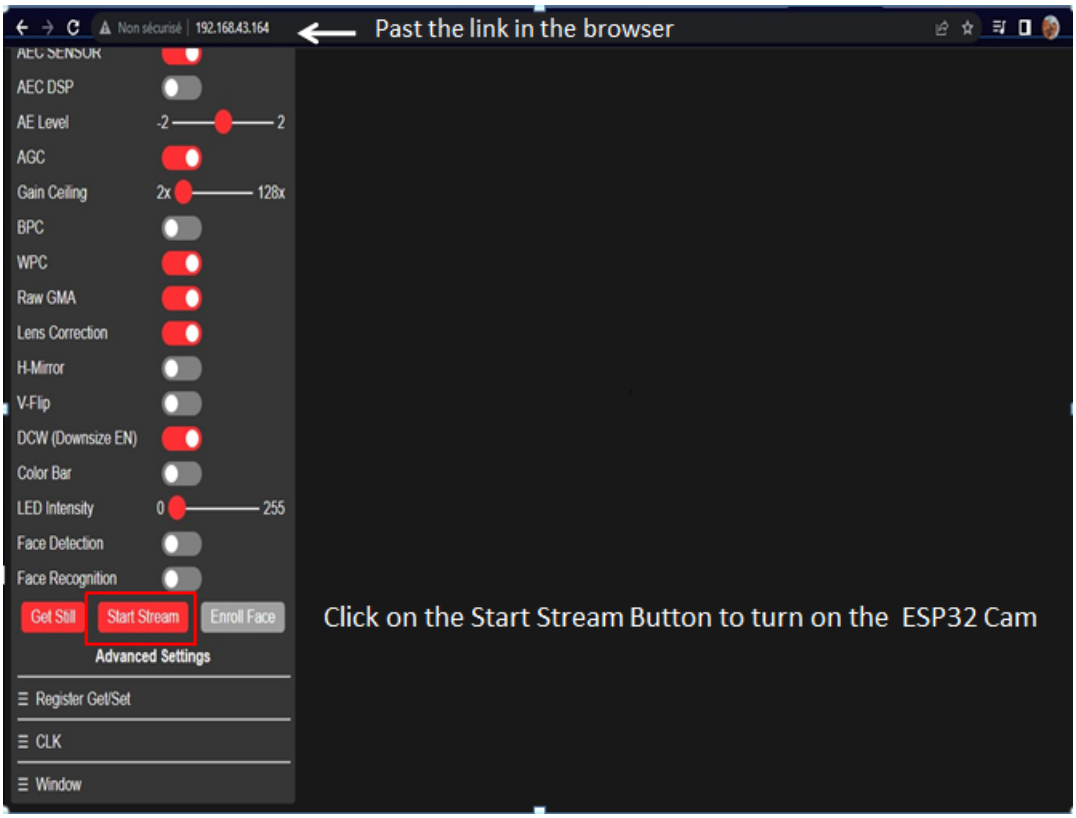


Figure 3.13: Past the link and turn on the ESP32 camera.

As a final step the ESP32 camera starts working and we can control it from the mobile phone (see figure 3.14 and figure 3.15):

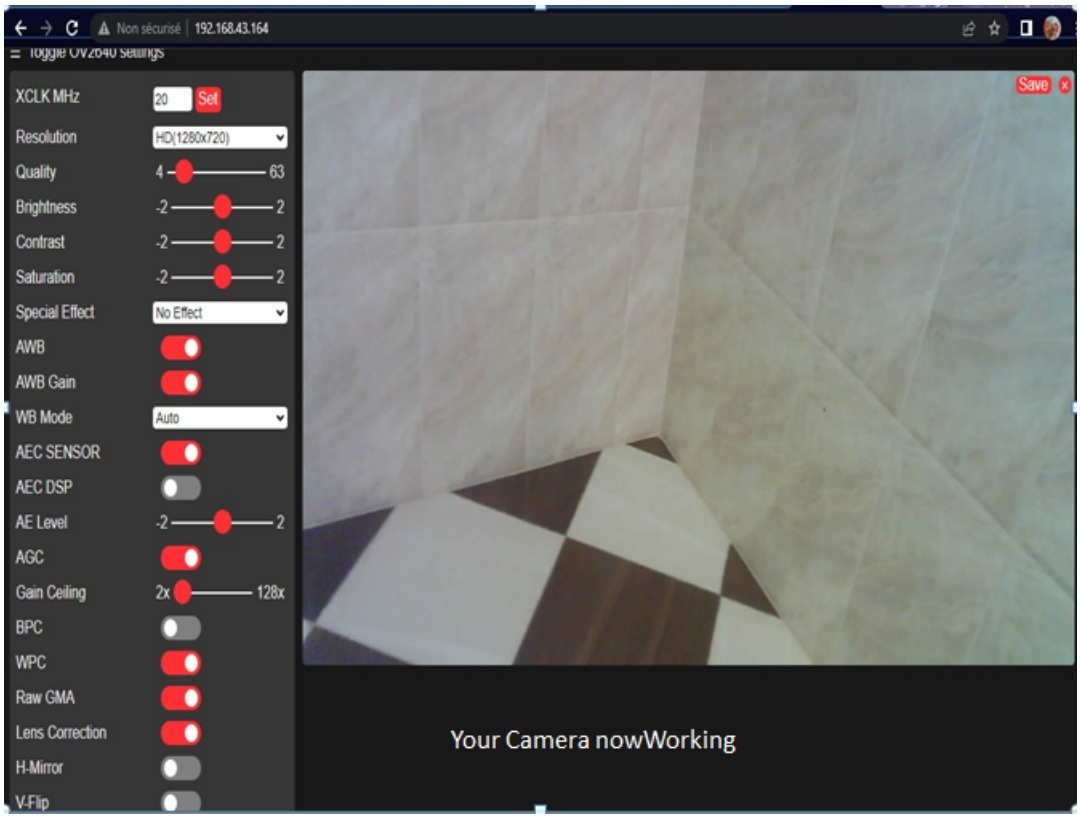


Figure 3.14: control camera from browser

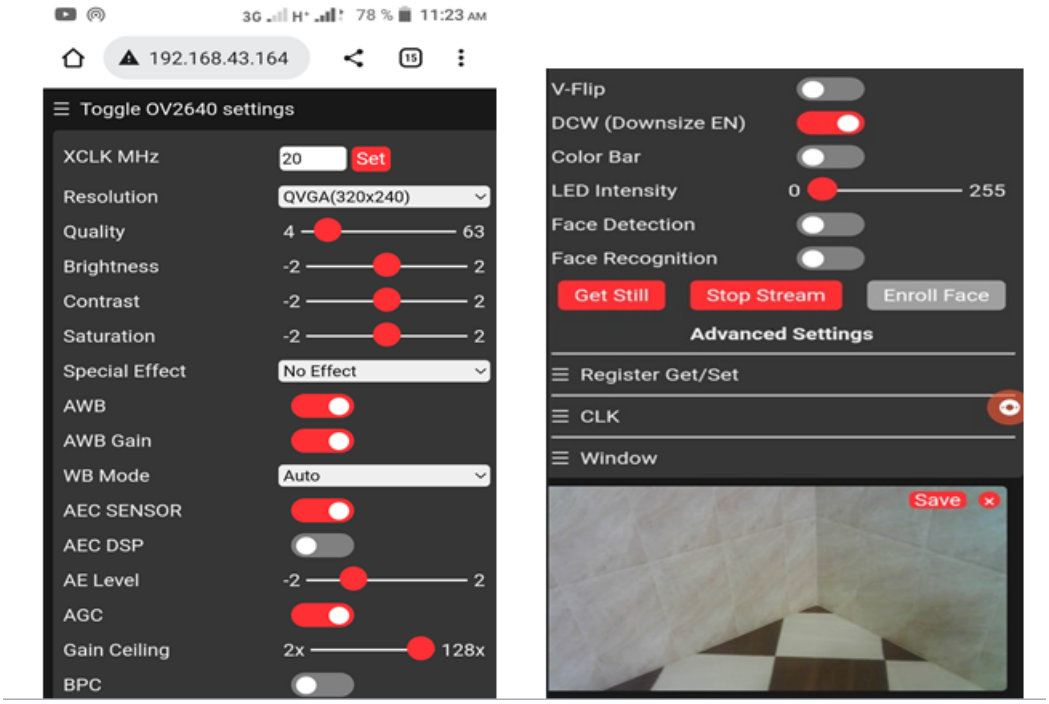


Figure 3.15: Control ESP32 camera from Mobile phone

3.3.2 Implementation of a Deep Learning model

1. Dataset preparation and preprocessing

The dataset has been used for the deep learning model is a custom dataset and in this section we have presented the main steps followed for the creation of the data:

- Data collection

The data collection is the first and the main step in the process of creating dataset; we have collected 8000 as a total number of images (4000 images for snakes and 4000 images for scorpions).

We must train our model on a high quality and perfect sized (not too big and not to small) images in order to give us high accuracy and more effective results and to reduce the percentage of mistakes. (see figure 3.16)

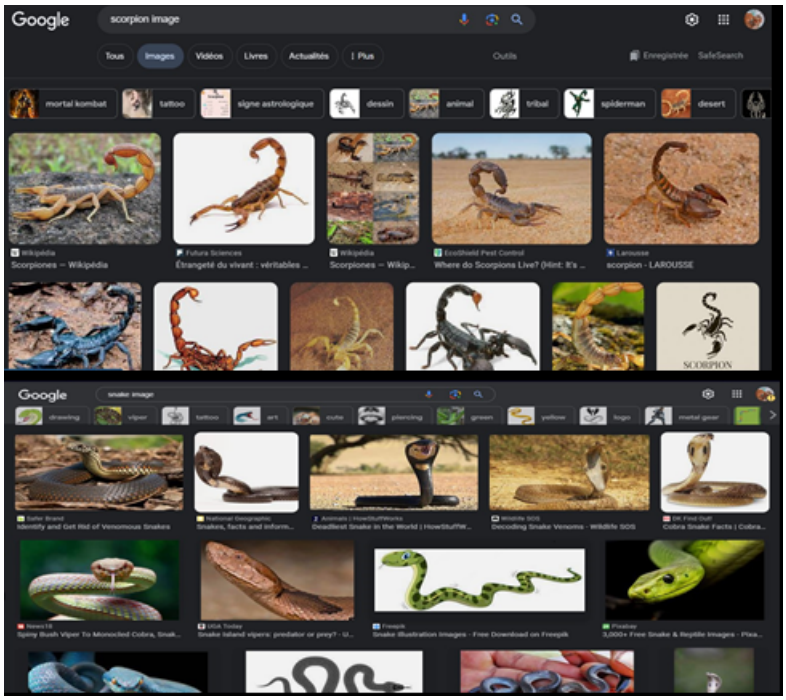


Figure 3.16: Collecting images.

After collecting the data we put them all in one folder (figure 3.17)

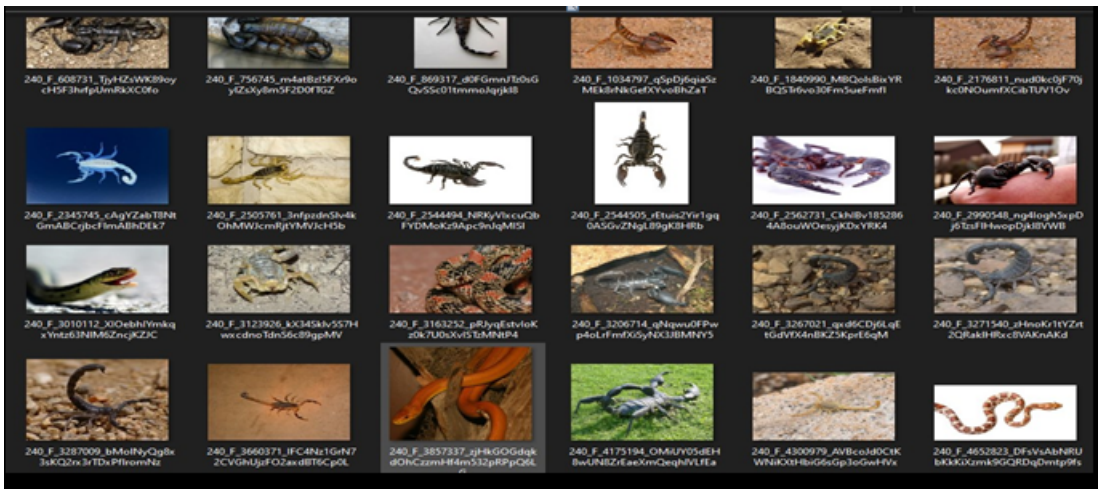


Figure 3.17: Collect the data in one folder.

Data annotation

After the data collection process we must annotate our images using CVAT

First, in CVAT we create a new task, we give the project name and the labels names (we have two labels snake and scorpion) the figure 3.18 shows the creation of the new task interface:

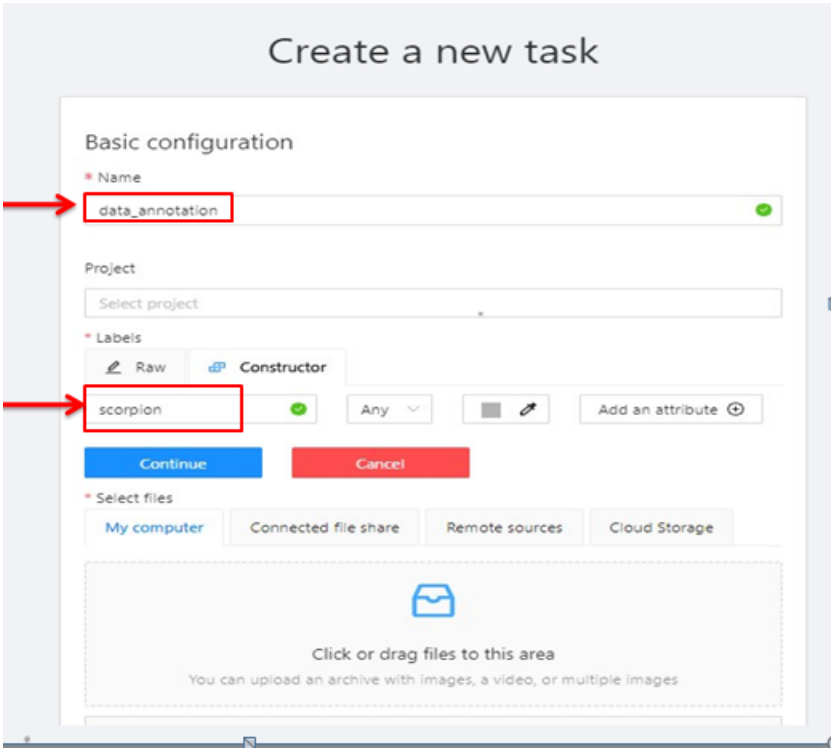


Figure 3.18: Launch New CVAT Task.

The figure 3.19 represents our labels classes

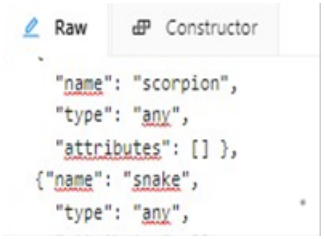


Figure 3.19: labels classes

After we create our task, we import the collected data folder (see figure 3.20)

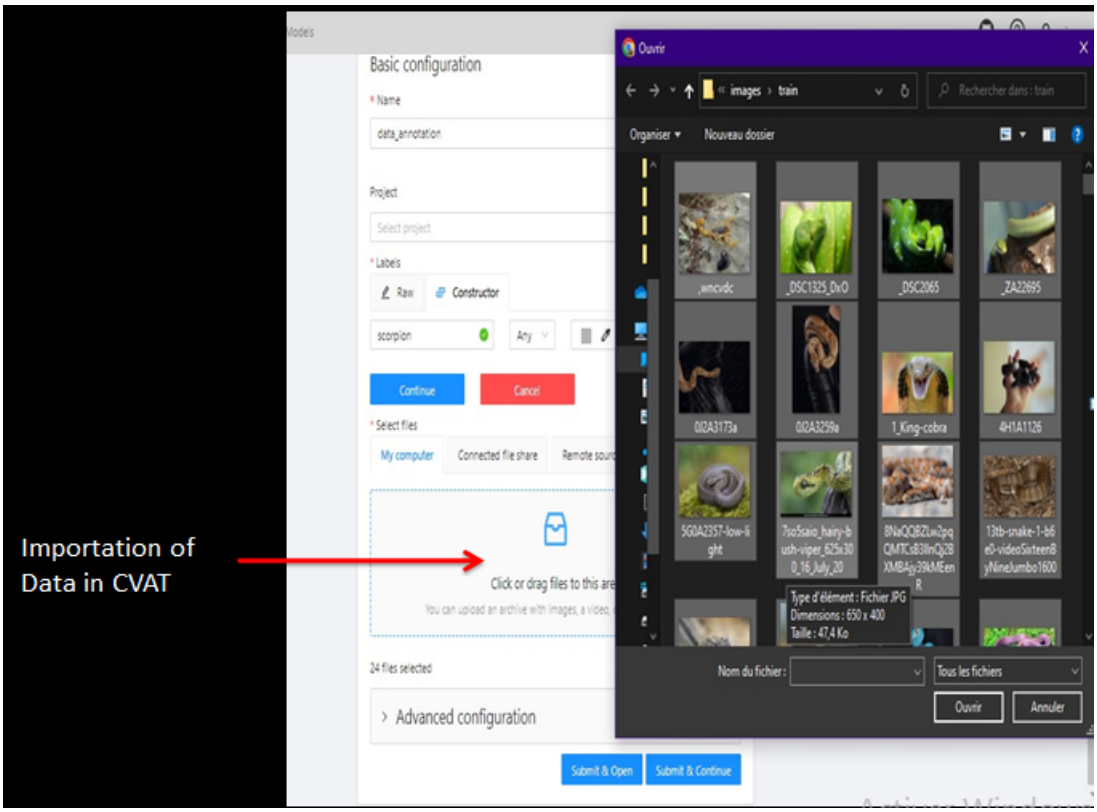


Figure 3.20: Import data in CVAT.

Once the data is uploaded, navigate back to tasks. From there, you will see a task page. In CVAT each label class is represented in a color (pink for snake and orange for scorpion), and with that we finish the task creation and the data is ready to annotate. (Details are shown in figure 3.21)

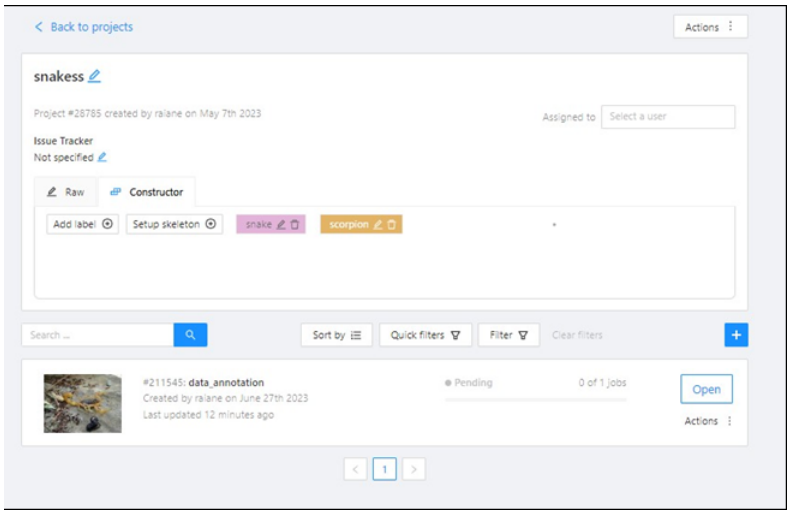


Figure 3.21: CVAT Task Page.

When we open the created task we will find the entire image we have imported are ready to annotate. Multiple types of shapes for annotation offered by CVAT: rectangle (bounding box), polygon, polyline, points, ellipse, cuboid, and tag.

In our case we have used the rectangle shape we put bounding boxes around the snake or the scorpion in the image (figure 3.22) and those bounding boxes must be precise.

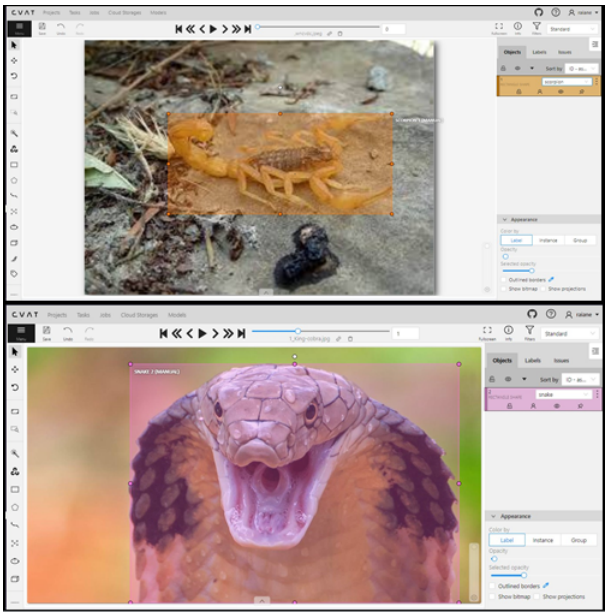


Figure 3.22: Annotated images with bounding boxes.

After finishing the annotation for all the snakes and scorpions images, we must export our

annotation from CVAT :

- first click "Save"(CVAT does not automatically save work.)
- Then click "Menu" you will see the following options (figure 3.23)

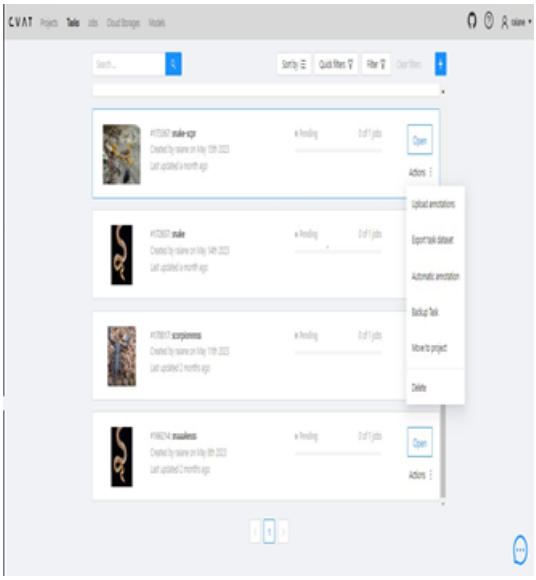


Figure 3.23: Menu from CVAT.

- Then click "Export task dataset" and you can choose among different formats. Because we are using YOLO model we choose the YOLO 1.1 format (figure 3.24)

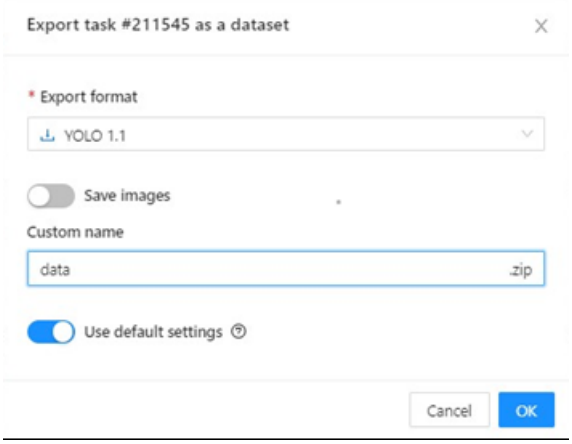


Figure 3.24: Export format.

The exported folder contains the text files for the annotations, we may notice that each image name is the same as each annotation text file name (figure 3.25). For this reason we must

organize our data so the model can learn correctly .

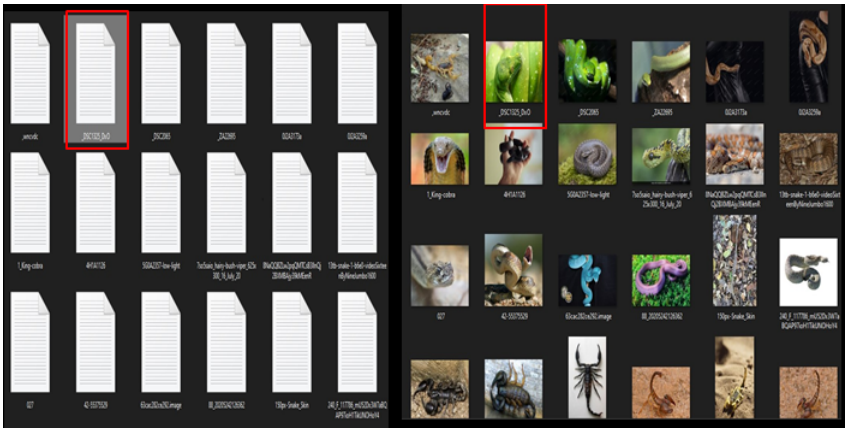


Figure 3.25: Exported folder

To organize our data we have created a folder , this folder contains two other folders one for images and one for the labels that we exported. We must take on consideration that the number of images is the same as the number of labels and each text file label has the same name as each image.

The figure 3.26 represents the annotation text file format:

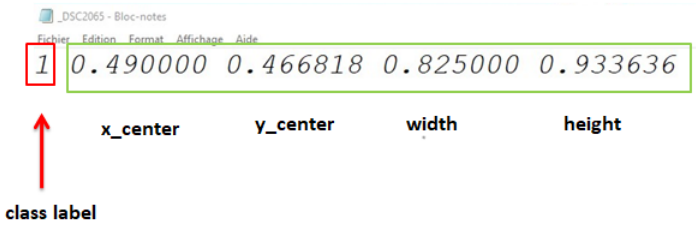
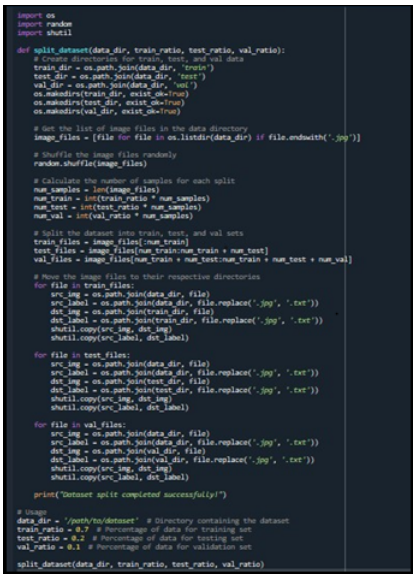


Figure 3.26: Annotation text file.

As the figure shows the text file contains class label each class is represented by a number (in our case 1 refers to snake and 0 refers to scorpion). The text file also contains the dimension of the bounding boxes.

Data splitting

We have split our dataset into 70 training, 20 validations and 10 tests. The code used for data splitting is represented in the figure 3.27



```
import os
import random
import shutil

def split_dataset(data_dir, train_ratio, test_ratio, val_ratio):
    # Create directories for train, test, and val data
    train_dir = os.path.join(data_dir, 'train')
    test_dir = os.path.join(data_dir, 'test')
    val_dir = os.path.join(data_dir, 'val')
    os.makedirs(train_dir, exist_ok=True)
    os.makedirs(test_dir, exist_ok=True)
    os.makedirs(val_dir, exist_ok=True)

    # Get the list of image files in the data directory
    image_files = [file for file in os.listdir(data_dir) if file.endswith('.jpg')]

    # Shuffle the image files randomly
    random.shuffle(image_files)

    # Calculate the number of samples for each split
    num_samples = len(image_files)
    num_train = int(train_ratio * num_samples)
    num_test = int(test_ratio * num_samples)
    num_val = int(val_ratio * num_samples)

    # Split the dataset into train, test, and val sets
    train_files = image_files[:num_train]
    test_files = image_files[num_train:num_train + num_test]
    val_files = image_files[num_train + num_test:num_train + num_test + num_val]

    # Move the image files to their respective directories
    for file in train_files:
        src_img = os.path.join(data_dir, file)
        src_label = os.path.join(data_dir, file.replace('.jpg', '.txt'))
        dst_img = os.path.join(train_dir, file)
        dst_label = os.path.join(train_dir, file.replace('.jpg', '.txt'))
        shutil.copy(src_img, dst_img)
        shutil.copy(src_label, dst_label)

    for file in test_files:
        src_img = os.path.join(data_dir, file)
        src_label = os.path.join(data_dir, file.replace('.jpg', '.txt'))
        dst_img = os.path.join(test_dir, file)
        dst_label = os.path.join(test_dir, file.replace('.jpg', '.txt'))
        shutil.copy(src_img, dst_img)
        shutil.copy(src_label, dst_label)

    for file in val_files:
        src_img = os.path.join(data_dir, file)
        src_label = os.path.join(data_dir, file.replace('.jpg', '.txt'))
        dst_img = os.path.join(val_dir, file)
        dst_label = os.path.join(val_dir, file.replace('.jpg', '.txt'))
        shutil.copy(src_img, dst_img)
        shutil.copy(src_label, dst_label)

    print("Dataset split completed successfully!")

# Usage
data_dir = '/path/to/dataset'  # Directory containing the dataset
train_ratio = 0.7  # Percentage of data for training set
test_ratio = 0.2  # Percentage of data for testing set
val_ratio = 0.1  # Percentage of data for validation set

split_dataset(data_dir, train_ratio, test_ratio, val_ratio)
```

Figure 3.27: the code for data splitting.

We can present the dataset architecture in the figure 3.28 bellow:

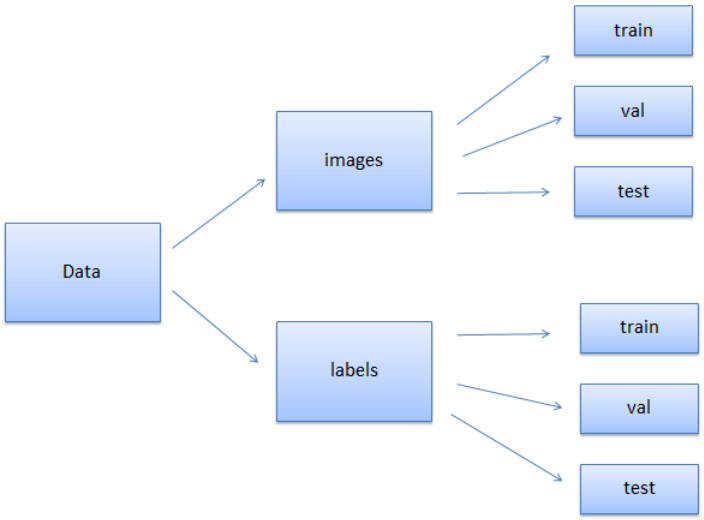


Figure 3.28: data architecture.

Data augmentation

We are using data augmentation techniques to increase the diversity and variability of our dataset. These techniques are including random rotations, flips (both vertical and horizontal), and adjustments to brightness. We use data augmentation to help us preventing overfitting.

The figure 3.29 presents the code for the data augmentation

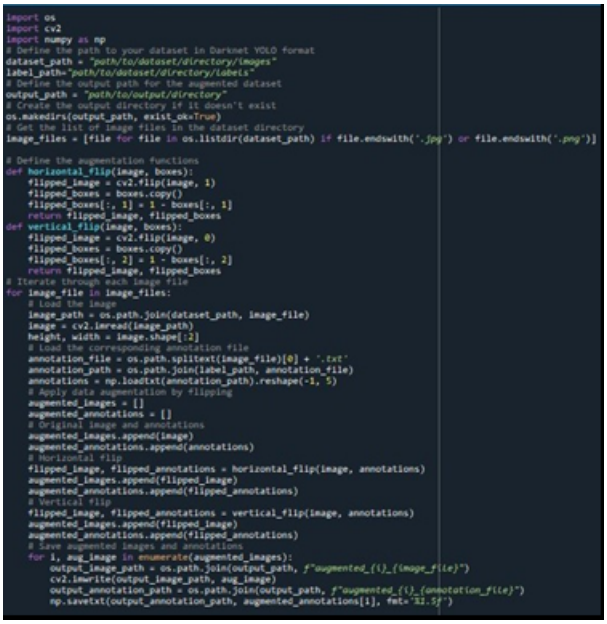


Figure 3.29: the code for data augmentation.

2. YOLOv8

YOLO is a popular real-time object detection algorithm known for its speed and accuracy. YOLOv8 refers to the eighth version of the YOLO object detection algorithm. YOLOv8 is an improvement over previous versions, incorporating advancements in deep learning techniques and network architecture.

YOLOv8 has been widely used in computer vision tasks such as object detection, instance segmentation, and even specialized domains like detecting specific objects (vehicles, pedestrians, or animals).

Versatility, accuracy, and real-time performance make yolov8 a choice for many computer vision applications.

We choose to train our model because for it is the fasts and the smallest YOLO versions the comparison is showed in figure 3.30

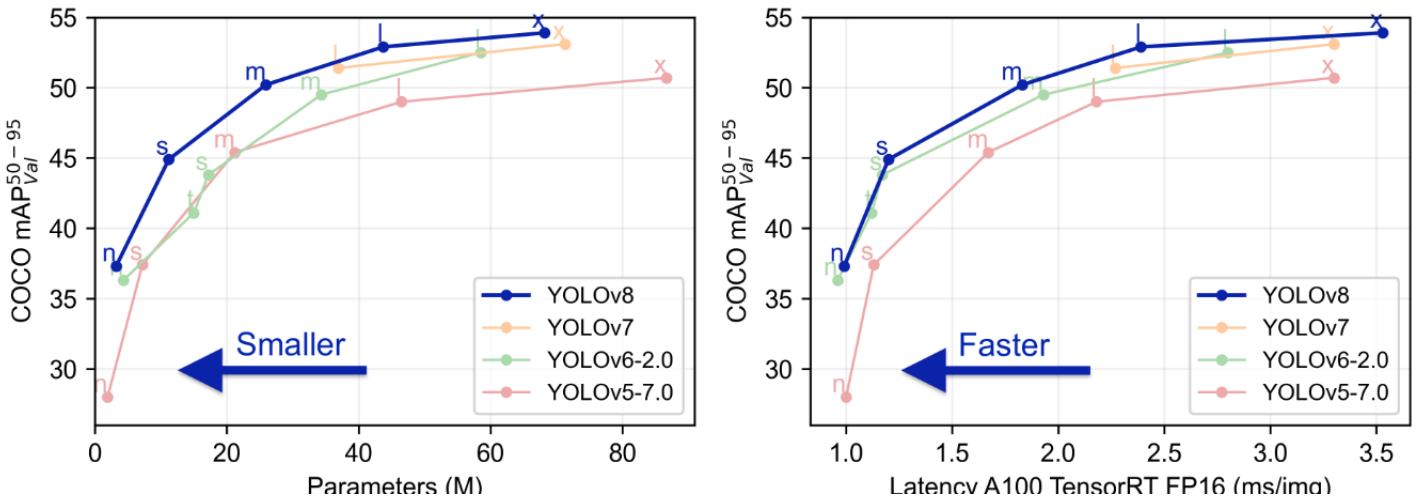


Figure 3.30: Comparison between YOLO model version.

YOLOv8 algorithm has five models represented in the figure 3.31.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 3.31: YOLOv8 models.

3. Training the YOLOv8 model

We trained our YOLO model in the hosted Jupyter notebook service COLAB, in this section we will explain all the steps of training our model with details.

The importation of the data folder in google drive is a necessary and require step before training the model. Next we must import the" dataset.yaml "file in google drive, which is commonly used to define the configuration and details of a dataset and provide a structured representation of dataset properties.(figure 3.32)

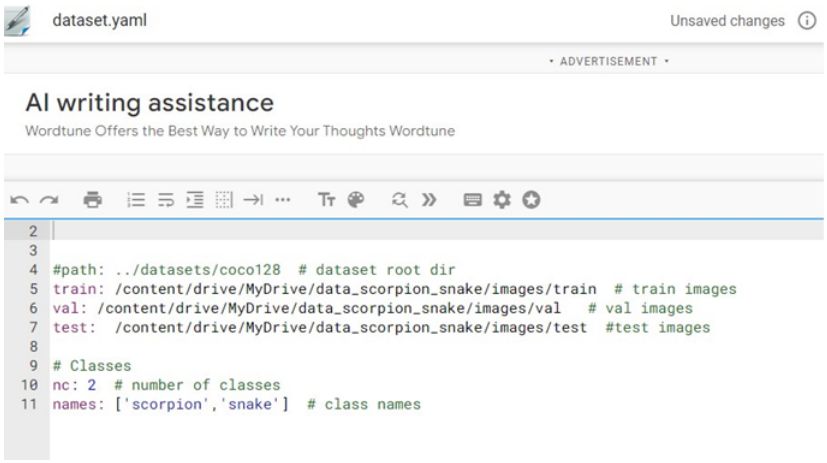


Figure 3.32: Dataset.yaml.

The ‘dataset.yaml‘ file contains information about the dataset such as:

• Dataset name: The name or identifier of the dataset (data scorpion snake).

• Data path: The directory or path where the dataset is stored

• Data format: The format of the data.

• Data labels.

• The labels or classes associated with the dataset, along with their corresponding numeric or string representations.

• Train-test-val split the division of the dataset into training, testing and validation subsets, including the path directories.

To start the model taining process in google colab, we need to run the code that mount google drive to the Colab environment.(figure 3.33)

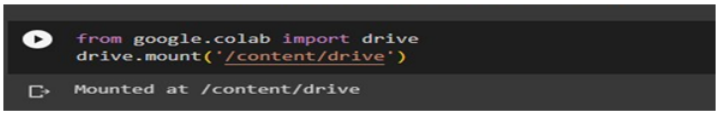


Figure 3.33: The code to mount google drive with colab.

By running this code, we will be able to access and interact with dataset.yaml and data directory imported in Google Drive from within the Colab notebook.

We have to install the ultralytics library in google colab notebook by using pip. (figure 3.34).

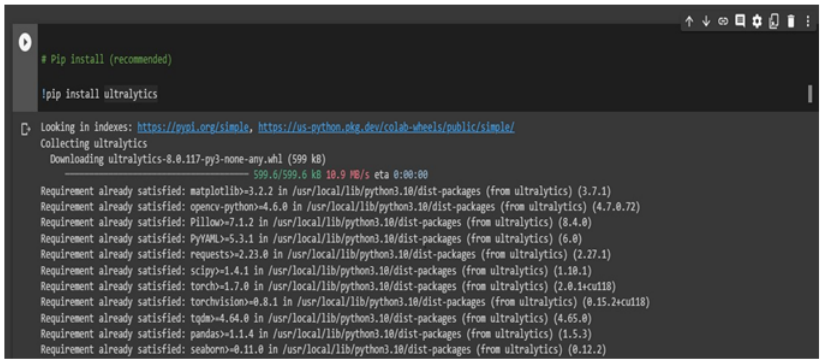


Figure 3.34: To install the ultralytics library in google colab notebook.

We can confirm that the ultralytics library has been imported correctly and check its version by running the code in the figure 3.35.

Figure 3.35: Confirm the importation of ultralytics library.

Confirming the ultralytics library can be useful for troubleshooting or ensuring that you have the desired version of the library installed. To train our task we run the code mentioned in the figure 3.36.

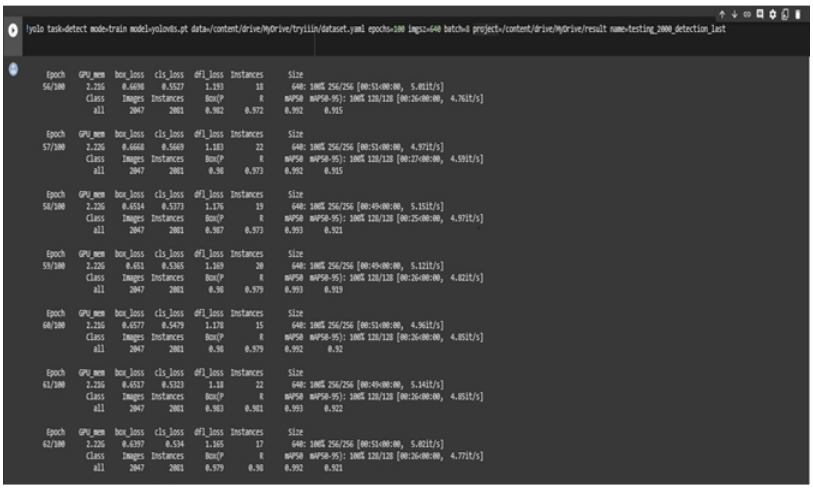


Figure 3.36: Training task.

The table below (3.1) presents the explanation of the training task code.

Parameter	Explanation
!	The exclamation mark (!) is used in Jupyter notebooks or Google Colab to execute shell commands.
Yolo	This is the command used to interact with the Ultralytics YOLOv8 library's command-line interface.
task=detect	This specifies the task to perform, which in this case is object detection.
mode=train	This indicates that the model will be trained.
model=yolov8s.pt	It specifies the model to be used for training, in this case, the yolov8s.pt model.
data=/content/drive/MyDrive /tryiiin/dataset.yam	This specifies the path to the dataset YAML file (dataset.yaml) that contains the configuration and details of the dataset to be used for training.
epochs=100	It sets the number of training epochs to 100, which determines how many times the entire dataset will be used for training.
imgsz=64	This sets the input image size for training to 640x640 pixels. The size of the input images affects the training process and detection performance.
batch=8	It sets the batch size to 8, which determines the number of images processed in each iteration during training. A larger batch size can speed up training but requires more memory.
project=/content/drive /MyDrive/result	This specifies the path to the project directory where the training results, such as trained models and logs, will be stored.
name=testing 2000 detection last	This sets the name for the training run, which will be used to identify the specific training session and its results within the project directory.

Table 3.1: explanation of the parameters of the training task code.

When the training is completed, we will test our model with images (figure 3.37).

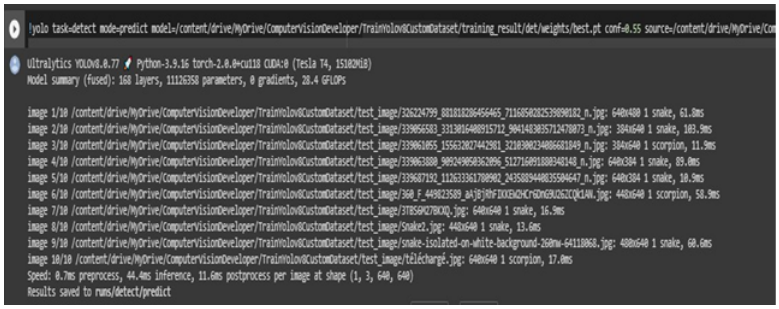


Figure 3.37: code of training the model with images.

The code explanation mentioned in the table 3.2 below.

!	The exclamation mark (!) is used in Jupyter notebooks or Google Colab to execute shell commands.
Yolo	This is the command used to interact with the Ultralytics YOLOv8 library's command-line interface.
task=detect	This specifies the task to perform, which in this case is object detection.
mode=predict	This indicates that the model will be used for inference or prediction.
model=/content/drive/MyDrive /ComputerVisionDeveloper /TrainYolov8CustomDataset /training result/det /weights/best.p	This specifies the path to the trained model (best.pt) that will be used for prediction. The model is located in the given path.
conf=0.55	This sets the confidence threshold for object detection to 0.55. Objects with detection confidence above this threshold will be considered as valid detections.
source=/content/drive/MyDrive /ComputerVisionDeveloper /TrainYolov8CustomDataset /test image	This specifies the source of the input data for prediction. It points to the directory or file where the test image(s) to be used for prediction are located.

Table 3.2: explanation of the parameters of the training task code.

We also tested our model with videos code mentioned in figure 3.38.

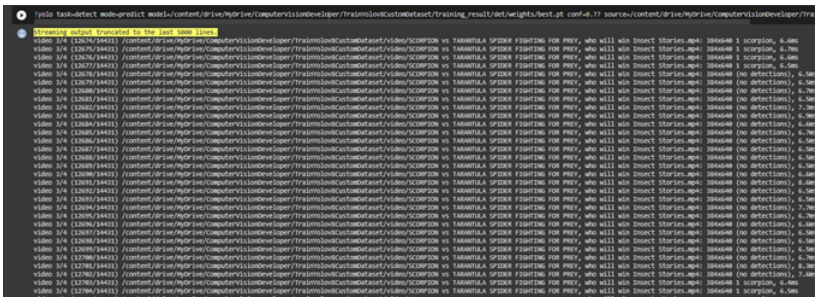


Figure 3.38: The code of training the model with videos.

4. Evaluation of Models

Model evaluation is a major step in assessing the performance and effectiveness of deep learning models, including object detection models. Before deploying the deep learning model in real life we have to evaluate it first. There are many different types of evaluation metrics available to evaluate the model like true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values are used to calculate variety of performance metrics including Accuracy, Precision, and F1score according to the following equations:

5. Upload the model of YOLOv8 in server

The server divides the video captured and send by the camera into frames. The server load the model and test it on each frame if the model detects a snake or scorpion the server will capture the image and send it to the homeowner. The code of uploading the model presented in figure 3.39

```
#Load the model of yolov8
def laod_model(frame):
    model = YOLO("path/to/best.pt")
    result=model(frame)[0]

    detections=sv.Detections.from_yolov8(result)
    if len(detections) > 0:

        for detection in detections:
            id_class=detection[3]
            x, y, w, h = int(detection[0][0]), int(detection[0][1]), int(detection[0][2] - detection[0][0]), int(detection[0][3]

            box(frame,x, y, w, h,id_class)
            turn_on_alarm()
            send_sms("There is a ",classes_name[classes_id]," in your House becarful please and get ride of it !!!")
            cv2.imwrite("pathto/captured_image.jpg", frame)
            image_path='pathof/captured_image.jpg'
            send_email_with_image(image_path)
            send_sms("Hi Becarful please there is a venomous animal in your house get ride ","\n Type of animal is ",classes_name
```

Figure 3.39: The code of uploading the YOLOv8 model to the server.

- Create load model function which takes a frame as a parameter.

- Give the path of the YOLO model.

- Draw boxes and set name on the detected scorpion or snake (figure 3.40).

- Turn on alarm (figure 3.43).

- Send sms (figure 3.41).

- Capture image.

- Get the path of captured image.

- Send email (figure 3.42).

The figure 3.40 below presented the draw and setting name on the detected scorpion or snake.

```
#Boxing
def box(frame, x, y, w, h,id_class):
    if(id_class==0):
      cv2.rectangle(frame, (int(x), int(y)), (int(x + w), int(y + h)), (0, 255, 0), 2)
      cv2.putText(frame, classes_name[id_class], (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    if(id_class==1):
       cv2.rectangle(frame, (int(x), int(y)), (int(x + w), int(y + h)), (0, 0, 255), 2)
       cv2.putText(frame, classes_name[id_class], (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)
```

Figure 3.40: Draw boxes and set name on the detected scorpion or snake code.

3.3.3 Send sms

When the model detects snake or scorpion, the server will send sms message to the homeowner mobile phone. The figure 3.41 presents the code of sending sms using the Twilio API.

```
#SEND_SMS

def send_sms(mssg):
    from twilio.rest import Client


    account_sid='account_sid'
    auth_token='password_sid'
    twilio_number='twilio number'
    my_phone_number='Homeowner number'
    client=Client(account_sid,auth_token)
    message= client.messages.create(
        body=mssg,
        from_=twilio_number,
        to=my_phone_number

        )
    print(message.body)
```

Figure 3.41: The code of send sms.

- def send.sms(mssg): a function named send.sms that takes one parameter mssg, which represents the message content to be sent.

- from twilio.rest import Client: imports the Client class from the twilio.rest module. It is used to interact with the Twilio API and send sms messages.

- account.sid = account sid' and auth token = 'password sid': These variables store the account SID and authentication token obtained from user's Twilio account. Each user must have a twilio account.

- twilio number = 'twilio number' and my phone number = 'Homeowner number' These variables store the Twilio phone number(generated by the twilio API) and the phone number of the homeowner.

- client = Client(account.sid, auth.token): This line creates an instance of the Client class using the account SID and authentication token.

- message = client.messages.create(body=mssg, from=twilio.number, to=my.phone.number): sends the sms message using the create method of the messages resource.

3.3.4 Send email

Homeowners receive an email message from the server if our model detects snake or scorpion. The email message contains an image of the snake or scorpion detected. Send email code is presented in the figure 3.42.

```
#SEND_EMAIL
def send_email_with_image(image_path):
    from email.message import EmailMessage
    import ssl
    import smtplib

    email_sender = 'email_sender'
    email_password = 'email_password'
    email_receiver = 'email_receiver'

    subject = 'Check out my message'
    body = """
   body message """

    # Create the email message object
    msg = EmailMessage()
    msg['From'] = email_sender
    msg['To'] = email_receiver
    msg['Subject'] = subject
    msg.set_content(body)

    # Add image attachment
    with open(image_path, 'rb') as f:
        image_data = f.read()
        msg.add_attachment(image_data, maintype='image', subtype='jpg', filename='image.jpg'

    # Send the email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(email_sender, email_password)
        smtp.sendmail(email_sender, email_receiver, msg.as_string())
        print("email sended")
```

Figure 3.42: The code of send email.

- First, we need to configure the security of the system email to allow accuse to send message with python code.

- Importation of the needed libraries.

- Create three variables to store the email sender, email password and email receiver.

- Generate the subject.

- Create the email message.

- Sending image with the generated message(add image attachement)

- Finally, send the email.

3.3.5 Launch alarm

The system turns on an alarm once a snake or scorpion detected (figure 3.43).

```
import pygame
import keyboard

def turn_on_alarm():
    # Initialize Pygame
    pygame.init()

    # Specify the audio file path
    file_path = "path/to/alarm.mp3"

    # Load the audio file
    pygame.mixer.music.load(file_path)

    # Play the audio file in an infinite loop
    pygame.mixer.music.play(-1)

    # Keep playing the audio until the Enter key is pressed
    while True:
        if keyboard.is_pressed("enter"):
            break

    # Stop the audio playback
    pygame.mixer.music.stop()

    # Quit Pygame
    pygame.quit()

turn_on_alarm()
```

Figure 3.43: The code of lunching alarm.

- Importation of the pygame and keyboard libraries.

- Create turn.on.alarm function.

- Initialization of pygame.

- Specify the audio file path.

- Keep the audio file playing until an enter key is pressed.

- Stop the audio playback.

- Quit paygame.

3.3.6 Start system

The code mentioned in the figure 3.44 below represented the start system code :

```
#Start Detector System
def start_Detector():
    # Open the video stream of Esp32
    stream_url = 'link of esp32 cam'
    cap = cv2.VideoCapture(stream_url)
    cv2.namedWindow("Cam Serveur Detector", cv2.WINDOW_NORMAL)
    cv2.resizeWindow("Cam Serveur Detector", 1000, 700)  # Set the desired window size
    cv2.moveWindow("Cam Serveur Detector", 50, 0)  # Set the top-left corner of the window

    # Check if the video stream of Esp32 cam is opened successfully
    if not cap.isOpened():
        print("Failed to open webcam")
        return

    # Read and display frames from the video stream of Esp32 cam
    while True:
        # Read a frame from the webcam
        ret, frame = cap.read()
        laod_model(frame)
        # If the frame was not successfully read, break the loop
        if not ret:
            break

        # Display the frame in a window called "Cam Serveur Detector"
        cv2.imshow("Cam Serveur Detector", frame)

        # Check for key press to exit
        if cv2.waitKey(1) == ord('q'):
            break

    # Release and close the window
    cap.release()
    cv2.destroyAllWindows()
```

Figure 3.44: The start system code.

3.4 Results

In this section we will present the obtained results from the YOLOv8 model training .

3.4.1 F1-Confidence curve

The curve presented in figure 3.45 is a F1-Cnofidence curve.

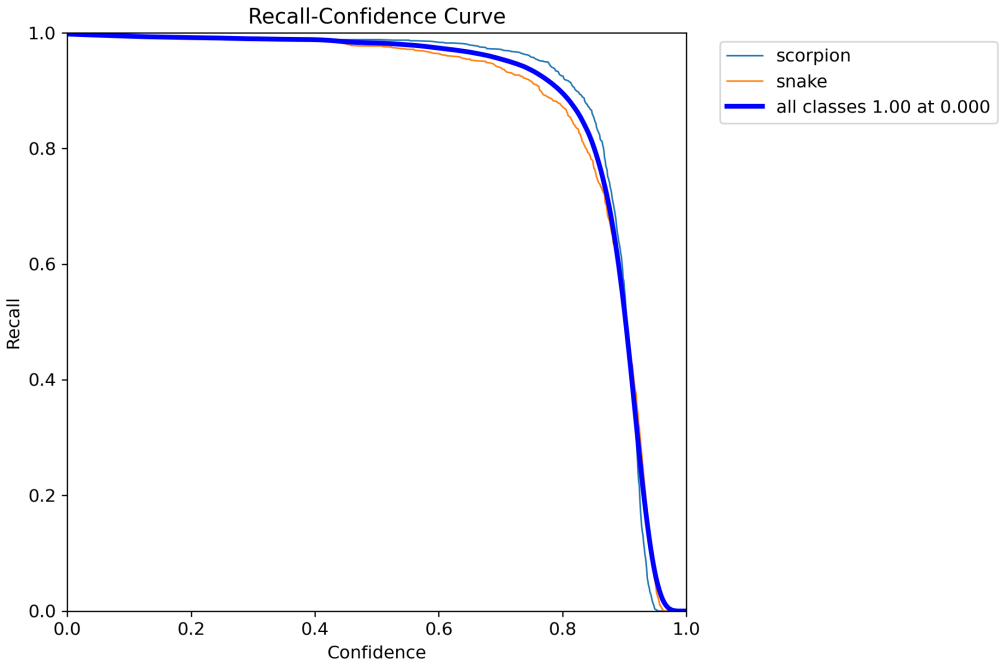


Figure 3.45: F1-Cnofidence curve.

F1-Confidence curve shows the relationship between the F1 score and the confidence level of the object detection model for two classes: scorpion and snake.

The F1 score indicates the model's accuracy, taking both precision and recall into account.

It represents the balance between correctly identifying positive instances (scorpions and snakes) and minimizing false positives and false negatives.

the curve indicates that the model achieves an F1 score of 0.98 for both scorpion and snake classes at a confidence level of 0.507. That means that the model achieves a high level of accuracy for detecting both scorpions and snakes, when the model is confident in its predictions (with a confidence level above 0.507).

3.4.2 Precision-Confidence curve

The curve presented in figure 3.46 is a Precision-Confidence curve.

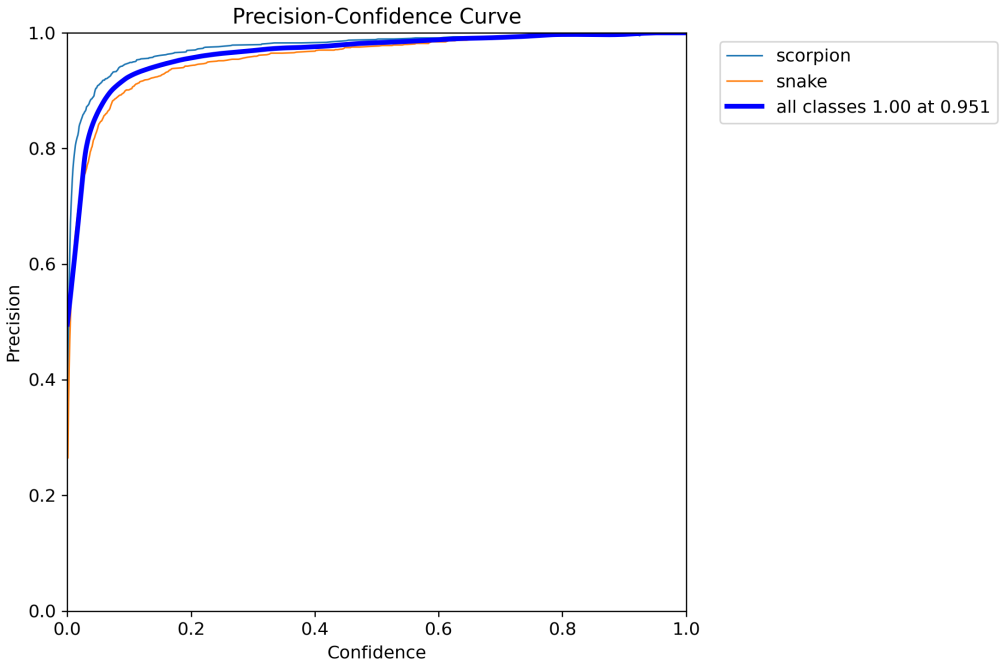


Figure 3.46: Precision-Confidence curve.

Precision-Confidence curve shows the relationship between the precision and confidence level of an object detection model for two classes: scorpion and snake. Precision is a measure of how accurately the model identifies true positives (correctly detected instances) compared to the total number of instances it labels as positive. It is an indication of the model's ability to avoid false positives.

The model achieves a precision of 1.00 for both scorpion and snake classes at a confidence level of 0.951. Which means that when the model is confident in its predictions (with a confidence level above 0.951), it demonstrates perfect precision, correctly identifying all instances of scorpions and snakes without any false positives.

3.4.3 Precision-Recall curve

The curve presented in figure 3.47 is a Precision-Recall curve.

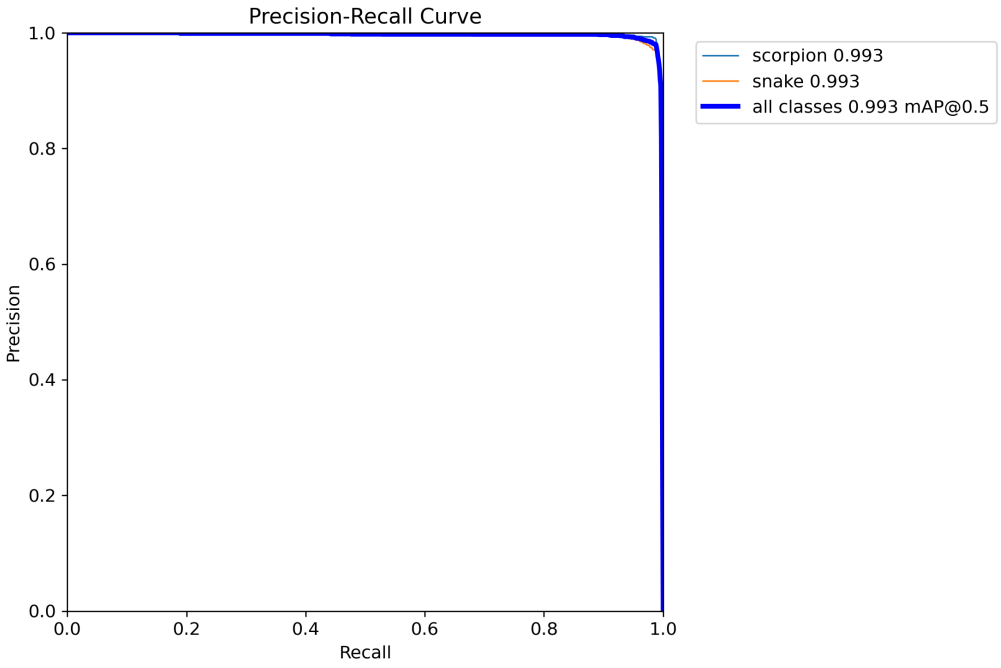


Figure 3.47: Precision-Recall curve.

Precision-Recall curve shows the trade-off between precision and recall for the classes scorpion and snake in an object detection model.

Precision is a measure of the model's ability to correctly identify true positives (correctly detected instances) in comparison to the total number of positive examples labeled. Recall, measures the model's ability to find all the positive instances in the dataset.

Our model achieves a high precision of 0.993 for both scorpion and snake classes. Which means when the model predicts an instance as positive, it is highly likely to be correct, with very few false positives. the recall for both classes is also 0.993, which indicate that the model effectively captures the majority of positive instances for scorpions and snakes.

3.4.4 Recall-Confidence curve

The curve presented in figure 3.48 is a Recall-Confidence curve.

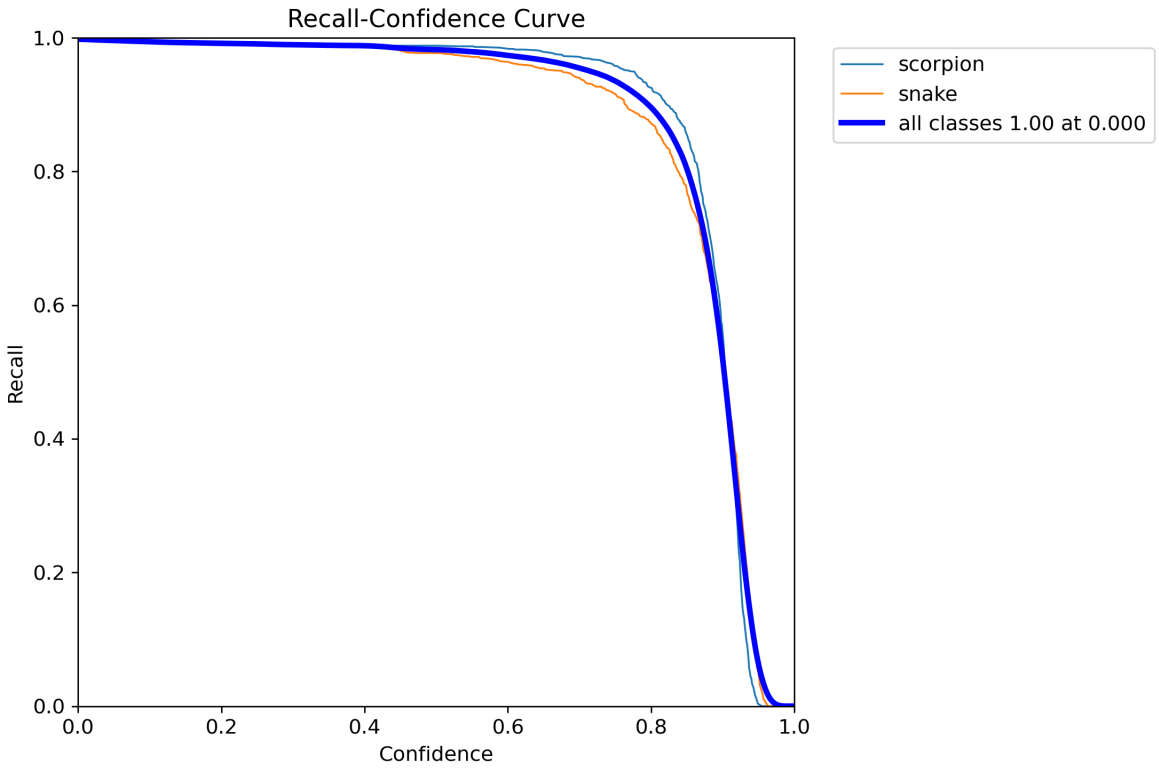


Figure 3.48: Recall-Confidence curve.

Recall-Confidence curve, shows the relationship between recall and confidence for the classes scorpion and snake in an object detection model.

The recall values for scorpion and snake classes are 1.00, which means that the model successfully detects all positive instances for both classes. This indicates a high level of performance in terms of capturing all instances of scorpions and snakes in the dataset.

3.4.5 Confusion matrix

The figure 3.49 is a represents the confusion matrix.

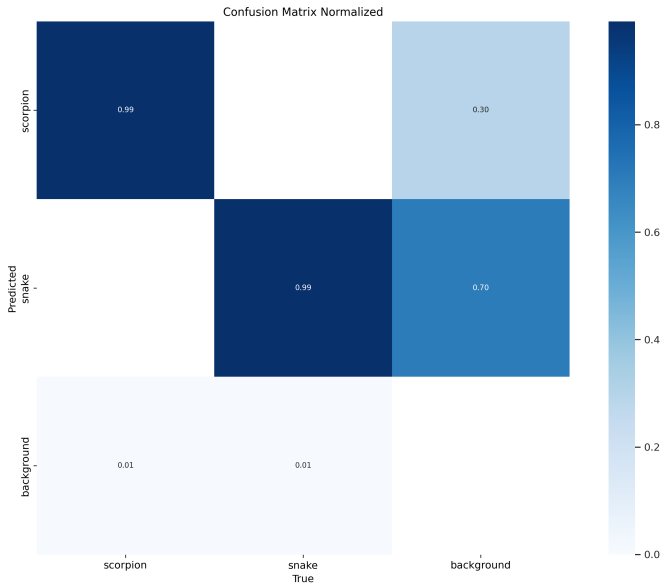


Figure 3.49: confusion matrix

The confusion matrix represents the classification results of an object detection model for three classes: background, snake, and scorpion. The values in the matrix are normalized, ranging from 0 to 1, indicating the proportion of instances that fall into a specific combination of true and predicted classes.

- The values along the diagonal represent the correctly predicted instances for each class. For example, the model correctly predicted 99 of snakes and 99 of scorpions.

- The values in the snake and scorpion rows indicate the instances that were wrongly predicted as snakes or scorpions, but they actually belong to the background class. For instance, 70 of the background instances were falsely classified as snakes, and 30 were falsely classified as scorpions.

- The values in the background column show the instances that were incorrectly classified as the background class but are actually snakes or scorpions. In this case, 80 of scorpions and 0 of snakes were misclassified as background

- The value in the top-left corner represents the true negatives, which are the instances correctly classified as the background class. In this case, 99 of the background instances were correctly identified.

The figure 3.50, shows the train batch0

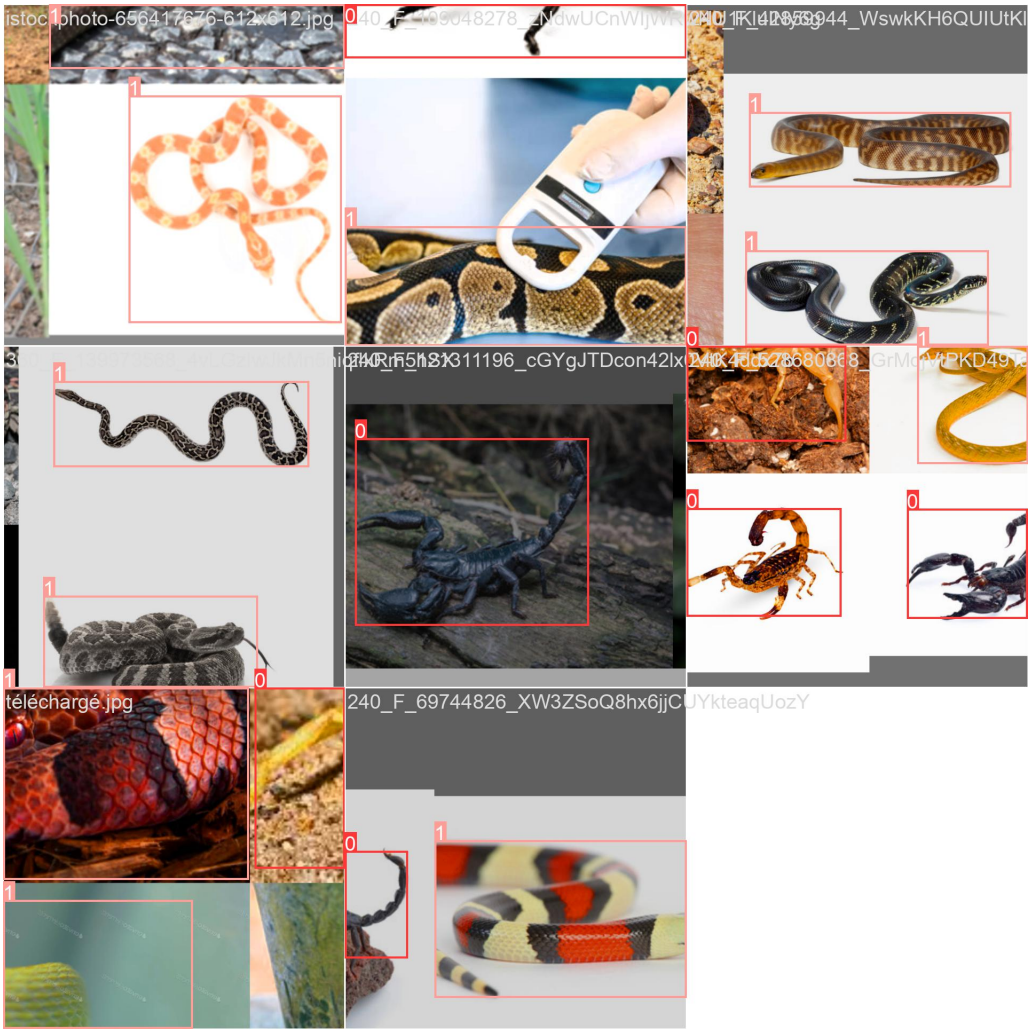


Figure 3.50: Train batch0

The figure 3.51, shows the train batch1

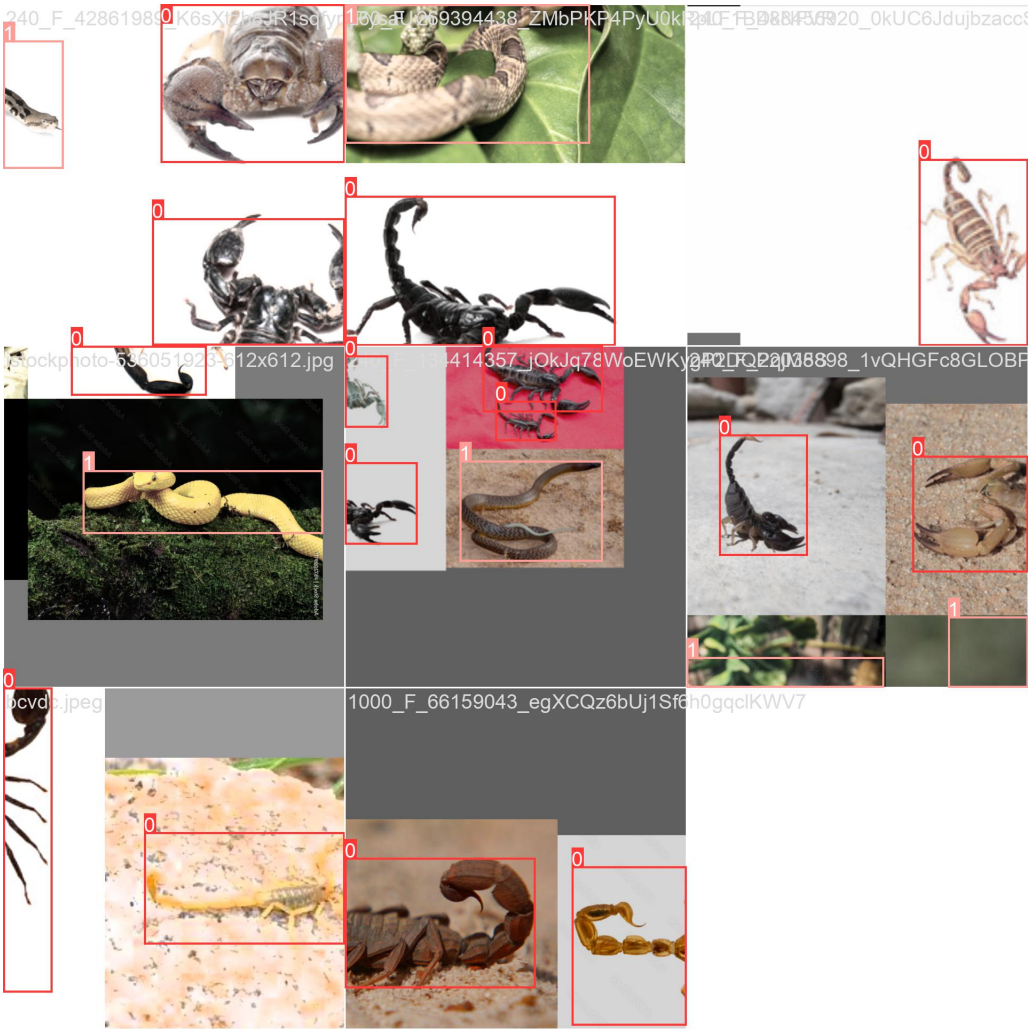


Figure 3.51: Train batch1

The figure 3.52, shows the train batch2

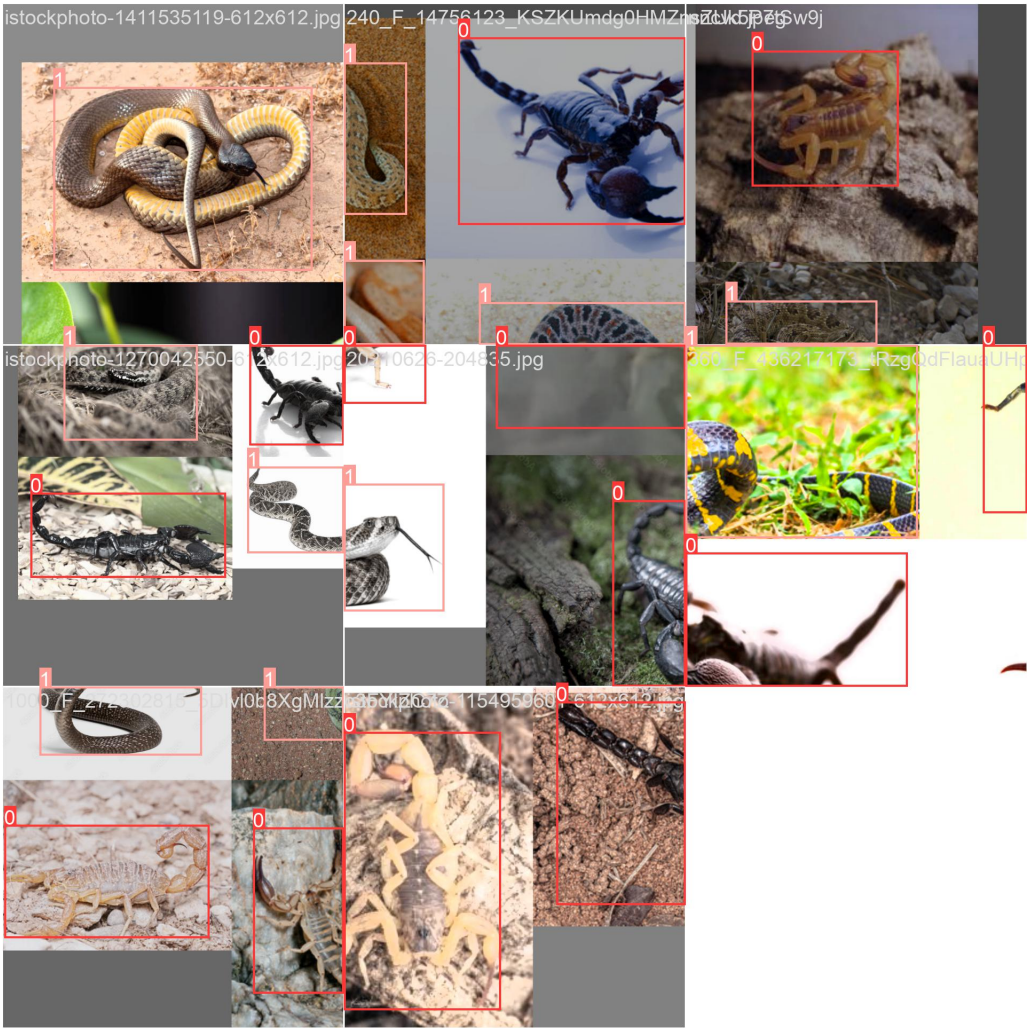


Figure 3.52: Train batch2

The figure 3.53, shows the validation batch 0 labels

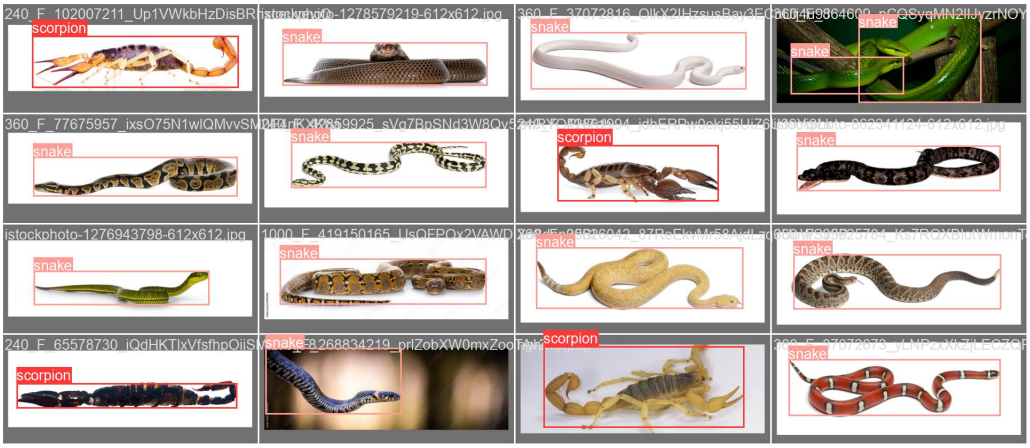


Figure 3.53: The val batch0 labels.

The figure 3.54, shows the validation batch 0 predict

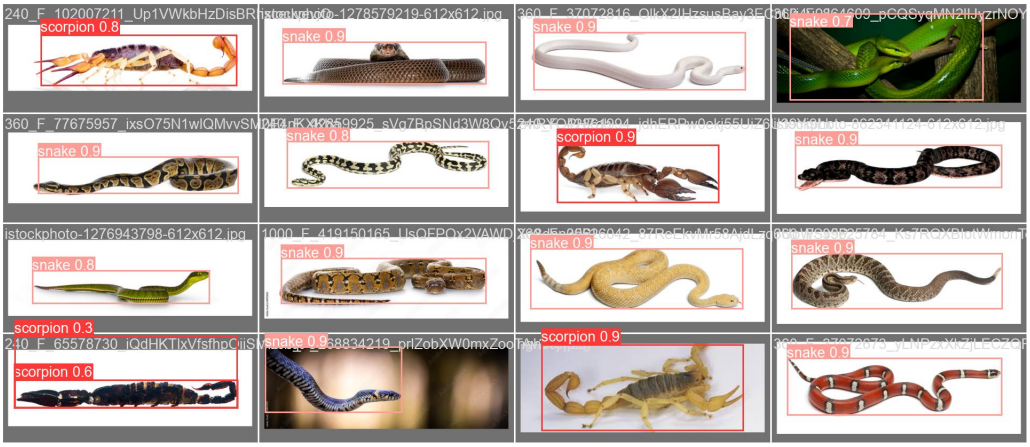


Figure 3.54: The val batch0 predict

3.5 Conclusion

In this chapter we have mentioned the tools, libraries and the frameworks we have utilized in our study. We have discussed the implementation of our system with detail. we have presented the results we obtained from training the YOLOv8 model.

General Conclusion and perspectives

Deep learning and object detection techniques have demonstrated immense potential in many fields, have achieved great success in recent years and have witnessed significant advancements especially in the field of animals detection. with the availability of large-scale datasets, the accuracy and efficiency of animal detection have greatly improved over time.

In this study we proposed a smart snakes and scorpion detection system intended for residents of desert areas. the system provides rapid and precise identification of snakes and scorpions and allows proactive measures to be taken, such as alerting residents, sending sms messages and emails accompanied by an image of the snake or the scorpion detected, to ensure the quick respond to the presence of snakes and scorpions in their surroundings and seeking professional assistance, thus preventing potential bites and minimizing the associated risks. YOLOv8 model is used for the snakes and scorpions detection. Our model is showing very excellent results and a high level of performance where the accuracy was 98 for both scorpion and snake classes.

In the near future, we are planing to add another device to our system, which is a drone. The drone will be able to catch and get rid of snakes and scorpions after they are detected.

References

[1] AI vs ML vs DL. https://insideaiml.com/blog/AI-vs-ML-vs-DL-1041?fbclid=IwAR2t9kHjHy5Sfl4JwqXvn5Pw1Oyoyu_5GjjInY_nfPcBltcQ5LSpmXt_SYo.

[2] Fritz AI. Object detection. https://www.fritz.ai/object-detection/, 2020. Accessed: Month Day, Year.

[3] Yasmeen Altujjar and Hala Mokhtar. Classification of smart home applications' requirements for the mac layer. 2018.

[4] Author. Ml: Relu and dropout layers, 2020. URL https://www.baeldung.com/cs/ml-relu-dropout-layers.

[5] Author. Cnn - introduction to pooling layer, 2020. URL https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[6] Author. Convolutional neural networks explained, 2020. URL https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[7] Author. Basic cnn architecture, 2020. URL https://www.upgrad.com/blog/basic-cnn-architecture/.

[8] Author. Basic cnn architecture, 2022. URL https://www.upgrad.com/blog/basic-cnn-architecture/.

[9] Author. Basic cnn architecture, Year. URL https://www.upgrad.com/blog/basic-cnn-architecture/.

[10] Jason Brownlee. A gentle introduction to transfer learning for deep learning. *Machine Learning Mastery*, 20, 2017.

[11] Jason Brownlee. Types of learning in machine learning, 2019. URL https://machinelearningmastery.com/types-of-learning-in-machine-learning/.

[12] Data Science Central. The artificial neural networks handbook - part 1, 2021. URL https://www.datasciencecentral.com/the-artificial-neural-networks-handbook-part-1/.

[13] Jin Cheng and Thomas Kunz. A survey on smart home networking. *Carleton University, Systems and Computer Engineering, Technical Report SCE-09-10*, 2009.

[14] Cisco. What is Wi-Fi? https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html?fbclid=IwAR3_mB-P8RPigJ4GhiFduCH8piu8thouanTb9KMKm-rigkjgxzTb6g2qCMw.

[15] Saturn Cloud. A comprehensive guide to convolutional neural networks - the eli5 way, 2020. URL https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/.

[16] Constellation Energy. What is a Smart Home? https://www.constellation.com/energy-101/what-is-a-smart-home.html?fbclid=IwAR3wWXFn2PpVjhqpJcsSB8gzrvOQeeJtMLd4lPK9x2LVS091g5iyTDzctvM.

[17] DeepAI. Deep learning - glossary and terms, 2021. URL https://deepai.org/machine-learning-glossary-and-terms/deep-learning.

[18] IBM Developer. Comparing machine learning and deep learning architectures, 2017. URL https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/.

[19] Encord. Yolo object detection guide. https://encord.com/blog/yolo-object-detection-guide/, 2021. Accessed: Month Day, Year.

[20] Experfy. Coding deep learning for beginners: Types of machine learning, 2018. URL https://resources.experfy.com/ai-ml/coding-deep-learning-for-beginners-types-of-machine-learning/.

[21] Experfy. Coding deep learning for beginners: Types of machine learning, 2020. URL https://resources.experfy.com/ai-ml/coding-deep-learning-for-beginners-types-of-machine-learning/.

[22] Experfy. Coding deep learning for beginners: Types of machine learning, 2020. URL https://resources.experfy.com/ai-ml/coding-deep-learning-for-beginners-types-of-machine-learning/.

[23] GeeksforGeeks. Clustering in machine learning, 2019. URL https://www.geeksforgeeks.org/clustering-in-machine-learning/.

[24] Francisco Luis Giambelluca, Marcelo A Cappelletti, Jorge Osio, and Luis A Giambelluca. Scorpion detection and classification systems based on computer vision and deep learning for health security purposes. *arXiv preprint arXiv:2105.15041*, 2021.

[25] José María Gutiérrez, Juan J Calvete, Abdulrazaq G Habib, Robert A Harrison, David J Williams, and David A Warrell. Snakebite envenoming. *Nature reviews Disease primers*, 3(1):1–21, 2017.

[26] IBM. Deep learning - overview, 2020. URL https://www.ibm.com/topics/deep-learning.

[27] JavaTpoint. Unsupervised machine learning, 2018. URL https://www.javatpoint.com/unsupervised-machine-learning.

[28] JavaTpoint. Supervised machine learning, 2020. URL https://www.javatpoint.com/supervised-machine-learning.

[29] JavaTpoint. Machine learning, 2020. URL https://www.javatpoint.com/machine-learning.

[30] JavaTpoint. Unsupervised machine learning, 2020. URL https://www.javatpoint.com/unsupervised-machine-learning.

[31] JavaTpoint. Supervised machine learning, 2021. URL https://www.javatpoint.com/supervised-machine-learning.

[32] Ajitesh Kumar. Different types of cnnarchitectures explained: Examples. *Retrieved from vitalflux. com: https://vitalflux. com/different-types-of-cnn-architectures-exp lained-examples/Weapon Detection in Surveillance Videos*, 195, 2021.

[33] V7 Labs. Yolo object detection. https://www.v7labs.com/blog/yolo-object-detection, 2020. Accessed: Month Day, Year.

[34] Ravi Manne and Sneha C Kantheti. Application of artificial intelligence in healthcare: chances and challenges. *Current Journal of Applied Science and Technology*, 40(6):78–89, 2021.

[35] MathWorks. Object detection. https://www.mathworks.com/discovery/object-detection.html, 2021. Accessed: Month Day, Year.

[36] Tiago DP Mendes, Radu Godina, Eduardo MG Rodrigues, João CO Matias, and João PS Catalão. Smart home communication technologies and applications: Wireless protocol assessment for home area network resources. *Energies*, 8(7):7279–7311, 2015.

[37] Rajeev Piyare and Seong Ro Lee. Smart home-control and monitoring system using smart phone. *ICCA, ASTL*, 24:83–86, 2013.

[38] Rosslin John Robles and Tai-hoon Kim. Applications, systems and methods in smart home technology: A. *Int. Journal of Advanced Science And Technology*, 15:37–48, 2010.

[39] Samsung. What is Bluetooth? https://www.samsung.com/uk/support/mobile-devices/what-is-bluetooth/?fbclid=IwAR11jaLUVX9oPlOwpUZOKHnBqFhX-xh3CTVO4fs4oj-fpH7s9aBZC18ju44.

[40] Section.io. Introduction to yolo algorithm for object detection. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/, 2020. Accessed: Month Day, Year.

[41] Security.org. What is a Home Security System? https://www.security.org/home-security-systems/what-is-a-home-security-system/?fbclid=IwAR3gbI1qIeu5UKmFxjpvFOr64tfl2v661p24D9ew6UK9RLK0ZGgClB3rsJc.

[42] S Selmane, H El Hadj, and L Benferhat. The impact of climate variables on the incidence of scorpion stings in humans in m'sila's province in algeria. In *Proceedings of the World Congress on Engineering*, volume 2014, pages 2–4, 2014.

[43] Spiceworks. What is ML? https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/?fbclid=IwAR06PoleXPfwHrcMTSkygGeji5uLgHOGLaT1o3FpQ3MrNg9oNHYPbf385T8.

[44] TechTarget. Smart home app (home automation app). https://www.techtarget.com/iotagenda/definition/smart-home-app-home-automation-app?fbclid=IwAR1paBuGZrYXOLn17yeq-FQWsmC2H3u62l4uMqd76mECLtf-d2wOVwT5fG8, .

[45] TechTarget. ZigBee. https://www.techtarget.com/iotagenda/definition/ZigBee?fbclid=IwAR3KfB23YES8OBl-XiAmWPb-BF4y2KLPjETuVTUlJhm9UMQq6EBTi7R_H8Q, .

[46] TechTarget. Z-Wave. https://www.techtarget.com/iotagenda/definition/Z-Wave?fbclid=IwAR2XW-YqFJUd-jrnh7MNXr7KPYZbJwj3YJ96kGNYEB22P9ZH2yXPvkJvYKU, .

[47] TechTarget. Machine Learning (ML). https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML?fbclid=IwAR06PoleXPfwHrcMTSkygGeji5uLgHOGLaT1o3FpQ3MrNg9oNHYPbf385T8, .

[48] Vitalflux. Different types of machine learning models algorithms, 2020. URL https://vitalflux.com/different-types-of-machine-learning-models-algorithms/.

[49] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[50] Zihan Yang and Richard Sinnott. Snake detection and classification using deep learning. 2021.