People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Larbi Tébessi University - Tébessa
Faculty of Exact Sciences and Natural and Life Sciences
Department: Mathematics and Computer Science

End of studies thesis
For obtaining the MASTER diploma
Field: Mathematics and Computer Science
Sector: IT
Option: networks and computer security

Theme

A Smart System to Detect Cheating in the Online Exam.

Presented by:
Khelif Mohamed Sayeb

In front of the jury:

Defense date: 21 / 06 / 2021

بسم الله الرحمن الرحيم

Thanks

We thank God the Almighty who gave us the will and the courage to do this work.

First, we would like to thank our supervisor: ***Dr. Merzoug Soltane*** for all his advice, support, help, and guidance.

We thank the jury members for agreeing to judge our work.

We thank all the teachers of the department, and the colleagues of our 2020-2021 master class.

We thank all the people who supported us for near or far in carrying out this work.

Dedications

I dedicate this modest work to:

My parents who did encourage me to get to the end path;

My mother source of my success and happiness. My father who encourage me and always sacrifices to see me succeed; my respect and deep feelings towards them, may god protect them for me.

To my dear brother and sister.

To all those who helped me to carry out this work of research.

To all my friends and comrades.

Abstract

The COVID-19 pandemic, caused the whole globe some serious downfall in terms of economy, even the day-to-day activities are being restricted due to enforcements like lockdown. Wish made restrictions on the exam process.

In this paper we aim to make a fully automated online cheat detection system capable of detecting whether a student is attempting to cheat or not during the process of the exam. This system will use deep learning techniques (AI) such as face recognition, sound detection, active-window detection. To note that the face recognition process will use a CNN based module for its high accuracy and stability.

Résumer

La pandémie de COVID-19 a causé au monde entier une grave chute en termes d'économie, même les activités quotidiennes sont restreintes en raison d'applications telles que le verrouillage. Qui a fait des restrictions sur le processus d'examen.

Dans ce papier, nous visons à créer un système de détection de triche en ligne entièrement automatisé capable de détecter si un étudiant tente de tricher ou non pendant le processus de l'examen. Ce système utilisera des techniques d'apprentissage en profondeur (IA) telles que la reconnaissance faciale, la détection de sons, la détection de fenêtre active. A noter que le processus de reconnaissance faciale utilisera un module basé sur CNN pour sa grande précision et sa stabilité.

الملخص

الملخص تسبب جائحة COVID-19 في حدوث بعض الانهيارات الخطيرة في العالم بأسره من حيث الاقتصاد ، حتى يتم تقييد الأنشطة اليومية بسبب إجراءات الإنفاذ مثل الإغلاق .جعلت قيودًا على عملية الامتحان.

نهدف في هذه الورقة العلمية إلى إنشاء نظام آلي بالكامل للكشف عن الغش عبر الإنترنت قادر على اكتشاف ما إذا كان الطالب يحاول الغش أم لا أثناء عملية الاختبار .سيستخدم هذا النظام تقنيات التعلم العميق مثل التعرف على الوجوه واكتشاف الصوت واكتشاف النافذة النشطة .ملاحظة أن عملية التعرف على الوجوه ستستخدم وحدة معتمدة على الشبكة العصبية التلافيفية CNN لدقتها العالية واستقرارها .

Acronyms

AI:	Artificial Intelligence.
CNN or Conv:	Convolutional Neural Network.
NN:	Neural network.
URL:	Linear rectified units.
DL:	Deep Learning.
MLP:	Multilayer perceptron.
GPU:	Graphic processing unit.
API:	Application-programming interface.
CPU:	Central processing unit.
FCL:	Fully connected layer.

Summary

Figure List

Tables List

General Introduction

General Introduction

The COVID-19 pandemic, caused the whole globe some serious downfall in terms of economy, even the day-to-day activities are being restricted due to enforcements like lockdown. However, technology, allowed the strings of knowledge to remain intact. Well, knowledge is available, through the right sources. However, how about the examination processes, or the test process; it has to be executed under a controlled environment, with supervision. How are we to conduct it when all the students are sitting at their homes and are not allowed stepping out to gather for anything like an exam, how are we going to move with this thing forward? Well, technology yet again gives us a solution; intelligent monitoring system to detect cheating in the exam is our only option for this. However, security authenticity during the exams is one of the most important key criteria for its success.

- How can we ensure that the right candidate is the one attempting an online exam?
- How do we ensure that no cheating or malpractice is happening during the online exam process?
- Is there any way where system can to do all that; confirm the test taker identity and preventing any cheating or malpractice during the examination process?

Intelligent monitoring system to detect cheating in the exam is the answer to all our questions. Intelligent monitoring is the technology that allows you to monitor the online exams for candidates in different locations of the earth. There are different parameters of intelligent monitoring such as audio monitoring, video monitoring, and image monitoring, and user screen monitoring. With the help of all these mechanisms, we can provide a secure, controlled and cheat-free environment.

This paper is divided into four chapters structured as follows:

- Chapter_1: - Artificial Intelligence and Deep Learning.
- Chapter_2: - Detection and Recognition methods.
- Chapter_3: - a smart system to detect cheating in the online exam.
- In the end, we have a general conclusion and perspective.

Chapter I: Artificial Intelligence (Deep Learning).

Chapter 1 Artificial Intelligence (Deep Learning).

1. Introduction.

In this chapter we present some notions that related to our work such as Artificial Intelligence (AI) its definition and history then we move to Deep Learning were we define it and Neural Networks, see some of its top Algorithms such as CNN and LSTM.

2. Artificial Intelligence (AI)

Artificial Intelligence (AI) started in the 1950s, when some people in the nascent field of computing began to wonder if computers could be made to "think". A concise definition of the field would be: "the effort to automate the intellectual tasks usually performed by humans". As such, AI is a broad field that encompasses machine learning and deep learning, but also includes many more approaches that do not involve any learning. The original chess programs, for example, were only about hard-coded rules made by programmers and did not qualify as machine learning. For quite a long time, many experts believed that AI at the human level could be achieved by having programmers handcraft a large enough set of explicit rules to manipulate knowledge. This approach is known as Symbolic AI and was the dominant AI paradigm of the 1950s and late 1980s. It peaked in popularity during the expert systems boom of the 1980s. Although symbolic AI proved appropriate for solving well-defined logical problems, such as playing chess, it was difficult to find explicit rules for solving more complex fuzzy problems, such as classification of images, recognition of speech and translation. A new approach has emerged to take the place of symbolic AI: "machine learning" **[1]**.

3. Historic.

The history of Artificial Intelligence (AI) began in antiquity, with myths, stories and rumors of artificial beings endowed with intelligence or consciousness by master craftsmen. The seeds of modern AI were planted by classical philosophers who attempted to describe the process of human thinking as the mechanical manipulation of symbols. This work culminated in the invention of the programmable digital computer in the 1940s, a machine based on the abstract essence of mathematical reasoning. This device and the ideas behind it inspired a handful of scientists to begin seriously discussing the possibility of building an electronic brain.

The field of AI research was founded at a workshop held on the campus of Dartmouth College during the summer of 1956. [1] Those who attended would become the leaders of AI research for decades. Many of them predicted that a machine as intelligent as a human being would exist in no more than a generation, and they were given millions of dollars to make this vision come true.

Eventually, it became obvious that they had grossly underestimated the difficulty of the project. In 1973, in response to the criticism from James Lighthill and ongoing pressure from congress, the U.S. and British Governments stopped funding undirected research into artificial intelligence, and the difficult years that followed would later be known as an "AI winter". Seven years later, a visionary initiative by the Japanese Government inspired governments and industry to provide AI with billions of dollars, but by the late 80s the investors became disillusioned and withdrew funding again. Investment and interest in AI boomed in the first decades of the 21st century when machine learning was successfully applied to many problems in academia and industry due to new methods, the application of powerful computer hardware, and the collection of immense data sets.

3.1. Dartmouth Conference 1956: the birth of AI.

The Dartmouth Conference of 1956[1] was organized by Marvin Minsky, John McCarthy and 2 senior scientists: Claude Shannon and Nathan Rochester of IBM. The proposal for the conference included this assertion: "every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it".[1] The participants included Ray Solomonoff, Oliver Selfridge, Trenchard More, Arthur Samuel, Allen Newell and Herbert A. Simon, all of whom would create important programs during the first decades of AI research.[1] At the conference Newell and Simon debuted the "Logic Theorist" and McCarthy persuaded the attendees to accept "Artificial Intelligence" as the name of the field.[1] The 1956 Dartmouth conference was the moment that AI gained its name, its mission, its first success and its major players, and is widely considered the birth of AI.[2] The term "Artificial Intelligence" was chosen by McCarthy to avoid associations with cybernetics and connections with the influential cyberneticist Norbert Wiener.[2]

The diagram in figure 1 summarize the history of AI;

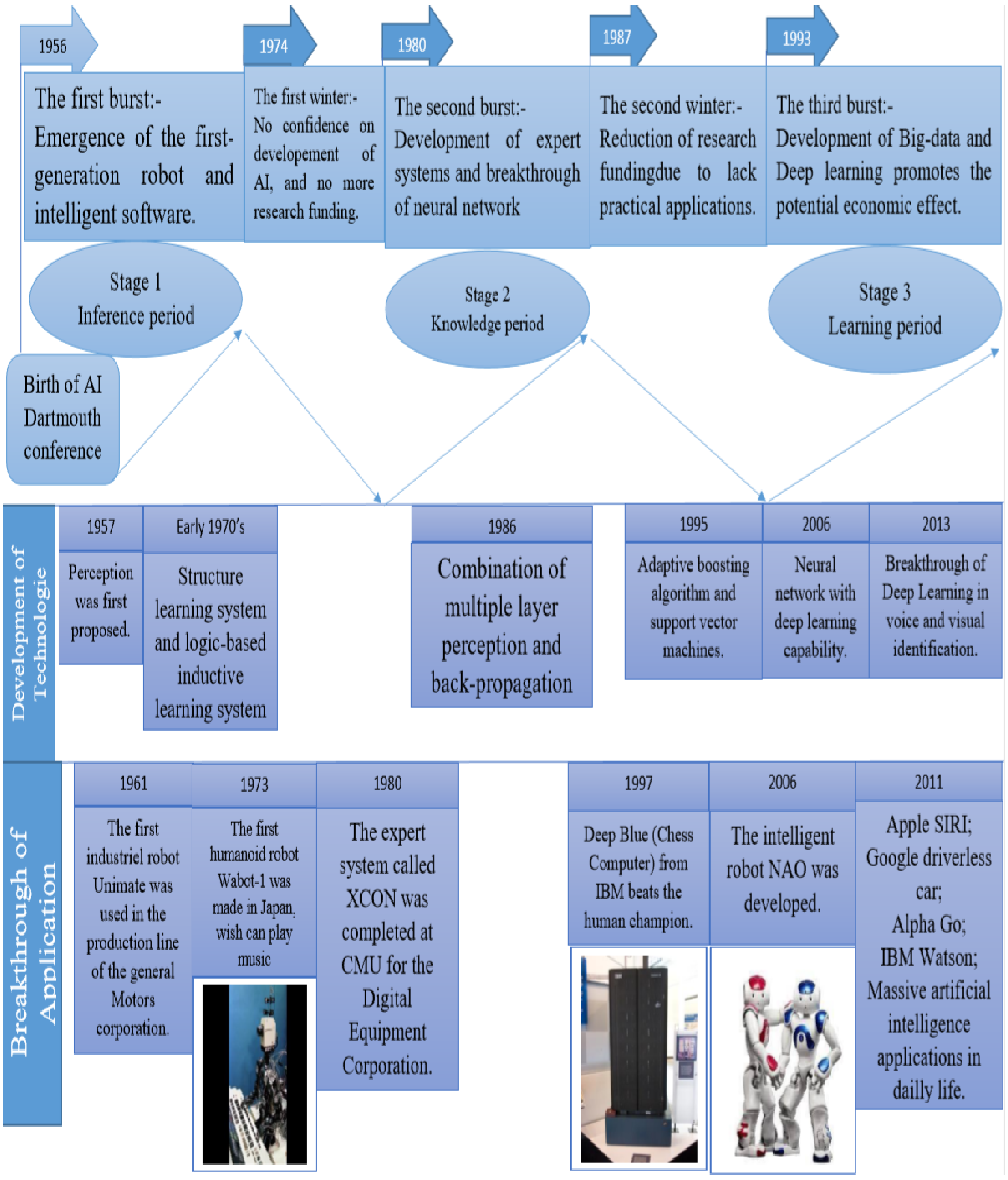

1956
The first burst:- Emergence of the first-generation robot and intelligent software.
1974
The first winter:- No confidence on developement of AI, and no more research funding.
1980
The second burst:- Development of expert systems and breakthrough of neural network
1987
The second winter:- Reduction of research fundingdue to lack practical applications.
1993
The third burst:- Development of Big-data and Deep learning promotes the potential economic effect.
Stage 1
Inference period
Stage 2
Knowledge period
Stage 3
Learning period
Birth of AI Dartmouth conference
Development of Technologie
1957
Perception was first proposed.
Early 1970's
Structure learning system and logic-based inductive learning system
1986
Combination of multiple layer perception and back-propagation
1995
Adaptive boosting algorithm and support vector machines.
2006
Neural network with deep learning capability.
2013
Breakthrough of Deep Learning in voice and visual identification.
Breakthrough of Application
1961
The first industriel robot Unimate was used in the production line of the general Motors corporation.
1973
The first humanoid robot Wabot-1 was made in Japan, wish can play music
1980
The expert system called XCON was completed at CMU for the Digital Equipment Corporation.
1997
Deep Blue (Chess Computer) from IBM beats the human champion.
2006
The intelligent robot NAO was developed.
2011
Apple SIRI; Google driverless car; Alpha Go; IBM Watson; Massive artificial intelligence applications in daily life.

Figure 1: AI history

4. Deep Learning.

Deep learning is a set of machine learning methods that play an important role in artificial intelligence. This approach has enabled several major breakthroughs where a program can draw conclusions on new data. Indeed, Deep learning is based on an artificial neural network which is made up of thousands of units (neurons), each of which receives and interprets information from the preceding layer, for example: for visual recognition, the first layers of 'units identify lines, curves, angles, etc., upper layers identify shapes, combinations of shapes, objects, contexts, etc.

These techniques have enabled significant progress thanks to the development of computers and large databases (big data). This concept was introduced to bring Machine Learning closer to its main objective, namely artificial intelligence. It also relates to machine learning algorithms that try to learn at several levels of representation to model complex relationships between data corresponding to different levels of abstraction. In addition, it has a good ability to extract features through raw data thanks to the multiple layers of processing which are composed of various transformations. Linear and nonlinear **[2, 10]**.

4.1. Definition.

Deep learning uses artificial neural networks to perform sophisticated computations on large amounts of data. It is a type of machine learning that works based on the structure and function of the human brain.

Deep learning algorithms train machines by learning from examples. Industries such as health care, e-commerce, entertainment, and advertising commonly use deep learning. **[S1]**

4.2. Defining Neural Networks.

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers **[S1]**:

- The input layer
- The hidden layer(s)
- The output layer

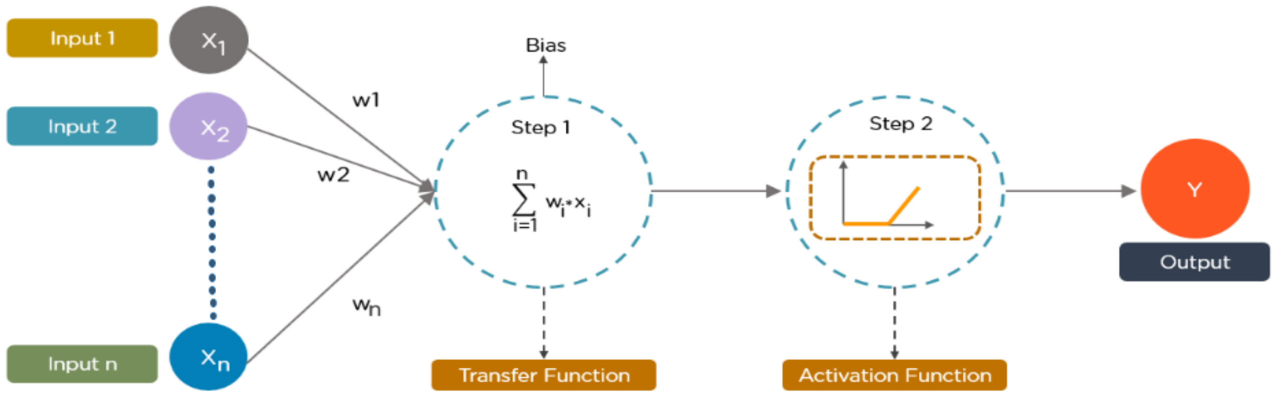


Figure 2 Neural Networks

Data provides each node with information in the form of inputs. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, nonlinear functions, also known as activation functions, are applied to determine which neuron to fire.

4.3. Deep Learning Algorithms.

4.3.1. Convolutional Neural Networks (CNNs).

CNN's, also known as ConvNets, consist of multiple layers and are mainly used for image processing and object detection. Yann LeCun developed the first CNN in 1988 when it was called LeNet. It was used for recognizing characters like ZIP codes and digits. [S1]

CNNs are widely used to identify satellite images, process medical images, forecast time series, and detect anomalies. [S1]

How Do CNNs Work?

CNN's have multiple layers that process and extract features from data:

Convolution Layer

- CNN has a convolution layer that has several filters to perform the convolution operation.

Rectified Linear Unit (ReLU)

- CNN's have a ReLU layer to perform operations on elements. The output is a rectified feature map.

Pooling Layer

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

- A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

Below is an example of an image processed via CNN.

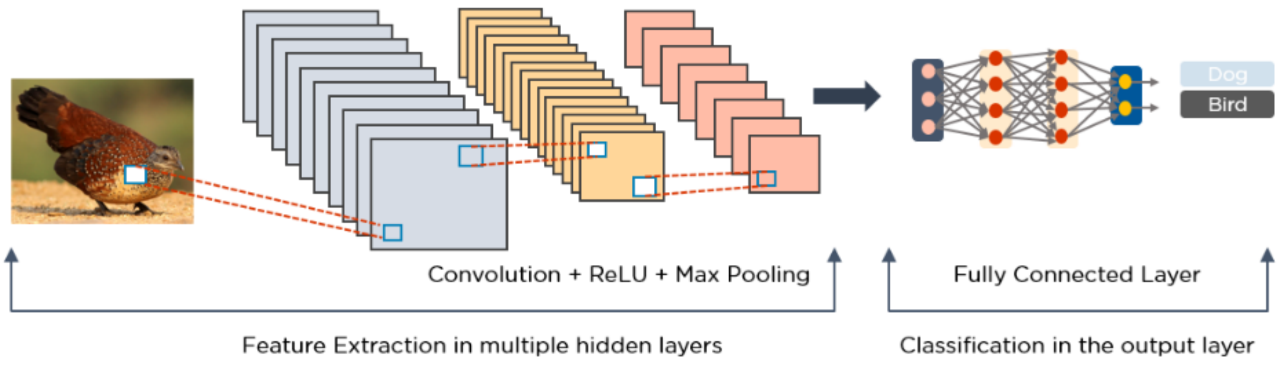

Dog
Bird
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

Figure 3 Convolutional Neural Networks [S1]

4.3.2. Long Short-Term Memory Networks (LSTMs).

LSTMs are a type of Recurrent Neural Network (RNN) that can learn and memorize long-term dependencies. Recalling past information for long periods is the default behavior. [S1]

LSTMs retain information over time. They are useful in time-series prediction because they remember previous inputs. LSTMs have a chain-like structure where four interacting layers communicate in a unique way. Besides time-series predictions, LSTMs are typically used for speech recognition, music composition, and pharmaceutical development. [S1]

How Do LSTMs Work?

- First, they forget irrelevant parts of the previous state
- Next, they selectively update the cell-state values

- Finally, the output of certain parts of the cell state

Below is a diagram of how LSTMs operate:

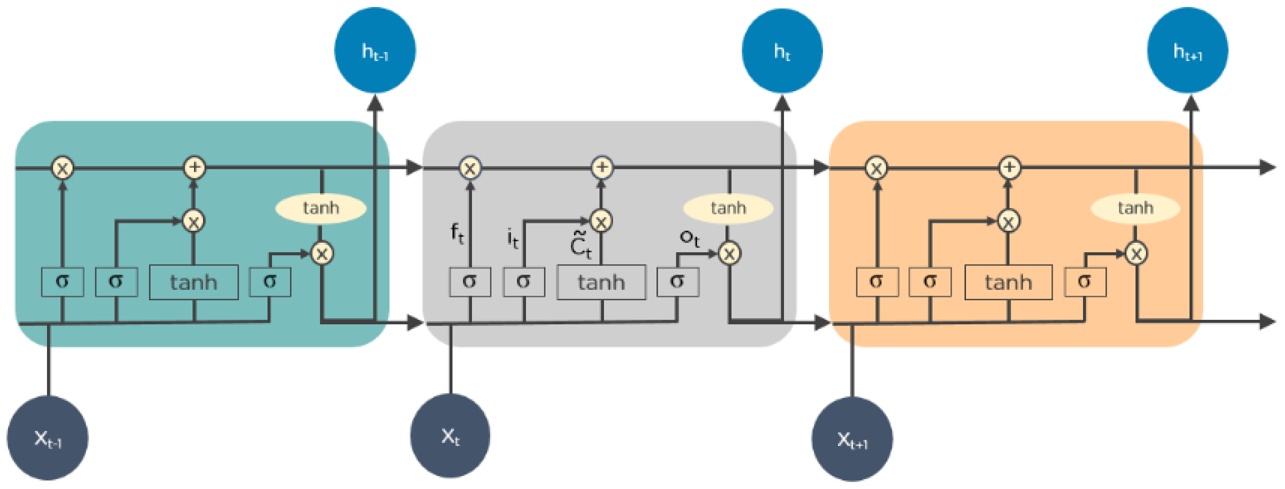


Figure 4 Long Short Term Memory Networks [S1]

4.3.3. Multilayer Perceptrons (MLPs).

MLPs are an excellent place to start learning about deep learning technology.

MLPs belong to the class of feedforward neural networks with multiple layers of perceptrons that have activation functions. MLPs consist of an input layer and an output layer that are fully connected. They have the same number of input and output layers but may have multiple hidden layers and can be used to build speech-recognition, image-recognition, and machine-translation software. [S1]

How Do MLPs Work?

- MLPs feed the data to the input layer of the network. The layers of neurons connect in a graph so that the signal passes in one direction.
- MLPs compute the input with the weights that exist between the input layer and the hidden layers.
- MLPs use activation functions to determine which nodes to fire. Activation functions include ReLUs, sigmoid functions, and tanhs.

- MLPs train the model to understand the correlation and learn the dependencies between the independent and the target variables from a training data set.

Below is an example of an MLP. The diagram computes weights and bias and applies suitable activation functions to classify images of cats and dogs.

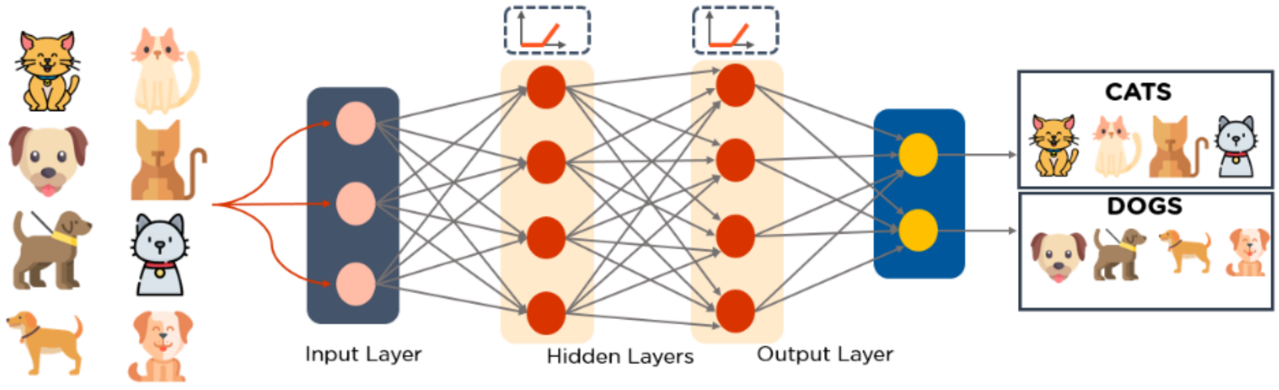


Figure 5 Multilayer Perceptrons [S1]

4.3.4. Autoencoders.

Autoencoders are a specific type of feedforward neural network in which the input and output are identical. Geoffrey Hinton designed Autoencoders in the 1980s to solve unsupervised learning problems. They are trained neural networks that replicate the data from the input layer to the output layer. Autoencoders are used for purposes such as pharmaceutical discovery, popularity prediction, and image processing. [S1]

How Do Autoencoders Work?

An Autoencoders consists of three main components: the encoder, the code, and the decoder.

- Autoencoders are structured to receive an input and transform it into a different representation. They then attempt to reconstruct the original input as accurately as possible.
- When an image of a digit is not clearly visible, it feeds to an Autoencoders neural network.
- Autoencoders first encode the image, and then reduce the size of the input into a smaller representation.

- Finally, the Autoencoders decodes the image to generate the reconstructed image.

The following image demonstrates how Autoencoders operate:

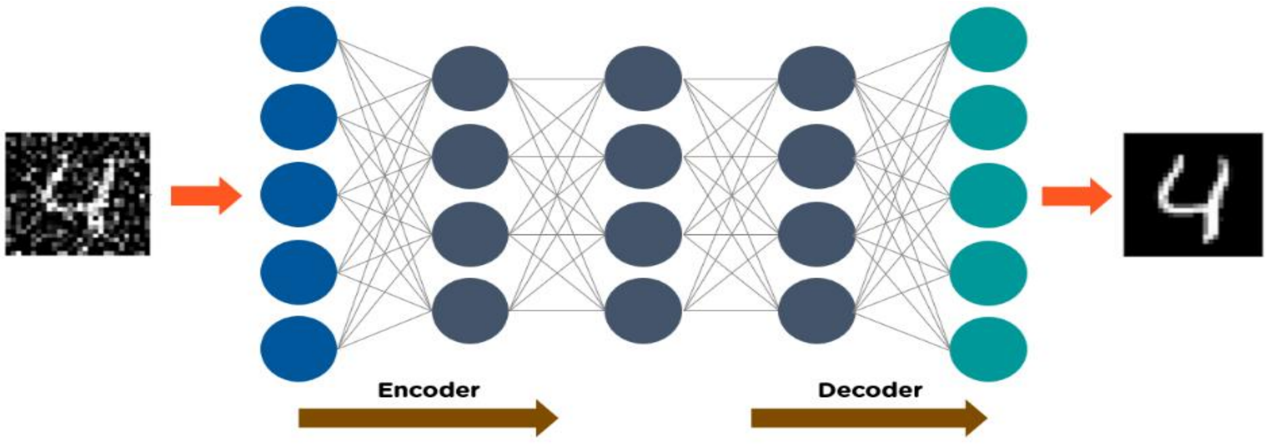


Figure 6 Autoencoders [S1]

5. Conclusion.

In this chapter we have seen that AI is intelligence demonstrated by machines, and what it can do from successfully understanding human speech, competing at the highest level in strategic game systems (such as chess and Go), and also imperfect-information games like poker, to self-driving cars. Also had read its history from antiquity, with myths, stories and rumors. The seeds of modern AI that were planted by classical philosophers who attempted to describe the process of human thinking as the mechanical manipulation of symbols, to Now were it still continue to grow. We also spoke of Deep Learning and some of its top algorithms like CNN and LSTM.

Chapter II: Intelligent Detection & Recognition Methods

Chapter 2 Intelligent Detection & Recognition Methods

1. Introduction

In this chapter we will talk about Object detection Methods it's definition and approaches, we also talk about object recognition methods what it is and also, its approaches, we take a look at face recognition plus face recognition techniques, Convolutional neural networks Architecture layers and Advantages, then we take a look at some related works to end it with a small conclusion.

2. Object detection Methods

Intelligent detection methods are a phenomenal field with various studies in our work we will focus on the object detection field. Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.[1] Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. It is widely used in computer vision tasks such as image annotation, [2] activity recognition, [3] face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a person in a video. Every object class has its own special features that helps in classifying the class – for example, all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

Methods for object detection generally fall into either neural network-based or non-neural approaches. For non-neural approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

- Non-neural approaches:
  - Viola–Jones object detection framework based on Haar features
  - Scale-invariant feature transform (SIFT)
  - Histogram of oriented gradients (HOG) features [5]
- Neural network approaches:
  - Region Proposals (R-CNN,[6] Fast R-CNN,[7] Faster R-CNN,[8] cascade R-CNN.[9])
  - Single Shot MultiBox Detector (SSD) [10]
  - You Only Look Once (YOLO) [11][12][13][4]
  - Single-Shot Refinement Neural Network for Object Detection (RefineDet) [14]
  - Retina-Net [15][9]
  - Deformable convolutional networks [16][17]

3. Object recognition methods

Object recognition – technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

- Approaches based on CAD-like object models
  - Edge detection
  - Primal sketch
  - Marr, Mohan and Nevatia[1]
  - Lowe

- Olivier Faugeras

- Appearance-based methods
  - Use example images (called templates or exemplars) of the objects to perform recognition
  - Objects look different under varying conditions:
    - Changes in lighting or color
    - Changes in viewing direction
    - Changes in size/shape
  - A single exemplar is unlikely to succeed reliably. However, it is impossible to represent all appearances of an object.
- Feature-based methods
  - A search is used to find feasible matches between object features and image features.
  - The primary constraint is that a single position of the object must account for all of the feasible matches.
  - Methods that extract features from the objects to be recognized and the images to be searched.
    - surface patches
    - corners
    - linear edges

4. Facial recognition

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

While initially a form of computer application, facial recognition systems have seen wider uses in recent times on smartphones and in other forms of technology, such as robotics. Because computerized facial recognition involves the measurement of a human's physiological characteristics facial recognition systems are categorized as biometrics.

Although the accuracy of facial recognition systems as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless process.[18] Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance and automatic indexing of images.[19] They are also used widely by law enforcement agencies.

4.1. Techniques for face recognition

While humans can recognize faces without much effort, [20] facial recognition is a challenging pattern recognition problem in computing. Facial recognition systems attempt to identify a human face, which is three-dimensional and changes in appearance with lighting and facial expression, based on its two-dimensional image. To accomplish this computational task, facial recognition systems perform four steps. First face detection is used to segment the face from the image background. In the second step the segmented face image is aligned to account for face pose, image size and photographic properties, such as illumination and grayscale. The purpose of the alignment process is to enable the accurate localization of facial features in the third step, the facial feature extraction. Features such as eyes, nose and mouth are pinpointed and measured in the image to represent the face. The so established feature vector of the face is then, in the fourth step, matched against a database of faces.[21]

5. Convolutional neural networks

5.1. Introduction

Convolutional neural networks are designed to work with grid-structured inputs, which have strong dependencies. Images are an example of data structured in a grid and which are generally well suited for this type of neural networks. Other forms of data like text, time series and sequences (which are sequential data) can be seen as special cases of grid structure data with various types of relationships between adjacent elements. The vast majority of applications of convolutional neural networks focus on image data, although these networks can also be used for all types of temporal, spatial and spatiotemporal data. The main feature of convolutional neural networks is the convolution operation which is a dot product operation between a set of grid-structured weights and similar grid-structured inputs [18].

5.2. CNN Architecture

5.2.1. Architectural principle of a CNN

A convolutional neural network is not just a deep neural network with many hidden layers. Rather, it is a deep network that simulates the functioning of the brain's visual cortex to recognize and classify images or videos, and to discover an object or even part of an image.

The concept and functioning of convolutional neural networks are different from other neural networks, in fact a convolutional neural network has two distinct parts with an input in which an image in the form of a two-dimensional pixel matrix (with 2 dimensions, black and white), or a color image with 3 dimensions (colors: red, green and blue).

The first part of a convolutional neural network is the convolutional part, which is used to extract the characteristics of the image. Then the image goes through the filter sequence file, or the winding core, which leads to the creation of a new image called convolution maps.

Usually, intermediate filters reduce the resolution of the image. Then, the feature maps are flattened into a feature vector to form the input data for the fully connected layer portion. The main role of this layer (fully connected) is to combine the characteristics contained in the vector of its input for the classification of images. The output of the CNN will be neurons which represent one neuron per class and the values obtained are generally normalized between 0 and 1 [2, 19]. Figure II.1 shows the typical CNN architecture.

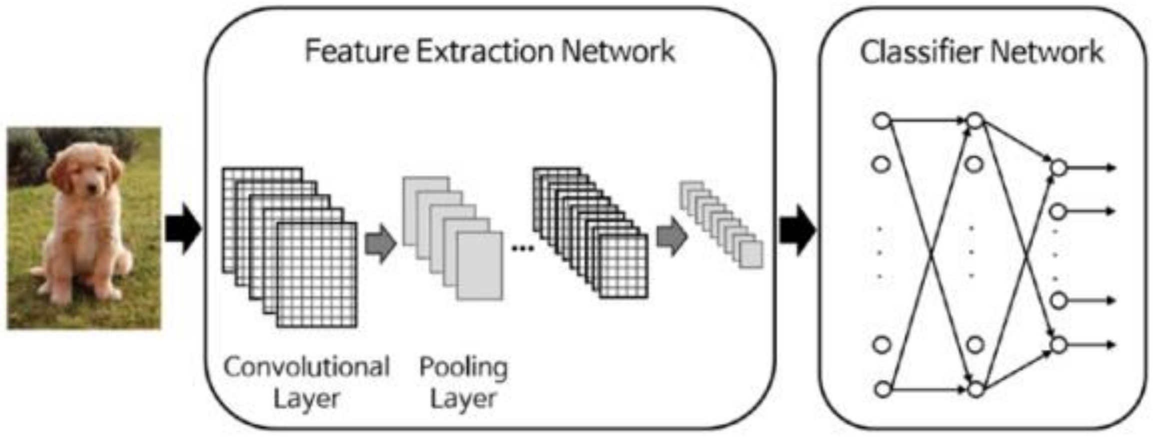


Figure 7 represents the typical architecture of CNN

5.2.2. The training of a new CNN

Creating a convolutional neural network is a difficult and expensive task because it requires good experience, hardware and the amount of data needed. The first step is to ground the network architecture i.e., the number of layers, the size and the matrix operations that connect them, then the training is to optimize the network parameters to reduce the error of output classification. Execution time can take several days for the best CNN networks because graphics processing units (GPUs) run on hundreds of thousands of frames **[19]**.

5.3. CNNs layers

CNN's have multiple layers that process and extract features from data:

1. **Convolution Layer**
   - CNN has a convolution layer that has several filters to perform the convolution operation.
2. **Rectified Linear Unit (ReLU)**
   - CNN's have a ReLU layer to perform operations on elements. The output is a rectified feature map.
3. **Pooling Layer**

- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

4. **Fully Connected Layer**
   - A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

5.4. Advantage of CNNs

The main advantage of convolutional networks is the use of the unique weight associated with the signals that all neurons enter into the same convolutional nucleus. This method reduces the memory footprint, improves performance, and enables processing stability.

6. Related works

6.1. Automated Online Exam Proctoring

Present a multimedia analytics system that performs automatic online exam proctoring. The system hardware includes one webcam, one wearcam, and a microphone for the purpose of monitoring the visual and acoustic environment of the testing location. The system includes six basic components that continuously estimate the key behavior cues: user verification, text detection, voice detection, active window detection, gaze estimation, and phone detection. By combining the continuous estimation components, and applying a temporal sliding window, the designed a higher-level feature to classify whether the test taker is cheating at any moment during the exam.

The paper was published by Yousef Atoum, Liping Chen, Alex X. Liu, Stephen D. H. Hsu, Xiaoming Liu Michigan State University, and East Lansing, MI, USA in 2017.

6.2. Examity Automated Proctoring

Examity is an online proctoring solution that gives students the flexibility to take exams remotely. It provides teachers, schools and students with the tools they need to prevent cheating and to preserve integrity.

7. Synthase

	Intelligent	User verification	Face recognition	Face detection	Voice detection	Gaze estimation	Iris detection	Active window detection	Recording
Automated Online Exam Proctoring	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No
Our Work	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Examity	No	Yes	No	NO	No	No	No	Yes	No

Table 1 comparison

8. Discussion

In the previous automated online exam proctoring system, there were many complaints from students and others that tried it like some cases were students were flagged by the system for simply reading the question with high volume or simply moving too much, in our work we try to provide a system which is more lenient while providing a decent accuracy in an attempt to provide a fair and well-shaped environment for the students partaking in the exams. As for Examity it cannot be called an automated system for it is a person who is responsible for watching during the exam and the system itself does not have even a little intelligence on the other hand our system work and decides on its own to give us a result in the end based on the data extracted using the webcam and the microphone of the user's computer.

9. Conclusion

In this chapter, we have presented some concepts that relate to our work, such as detection and identification of whether a person is known or not from images, and to track his or her movements. We have also presented the basic concepts of networks of convolutional neurons (CNN) and we have presented the basic operations of CNN networks namely, the operation of the convolutional layer, Polling, the fully connected layer and the ReLU activation function). We have seen related works such as Automated Online Exam Proctoring and Examity we also had a table of comparison with our system (Work to be).

In the next chapter, we will discuss our work and its details.

Chapter III: A Smart System to Detect Cheating in The Online Exam.

Chapter 3 A Smart System to Detect Cheating in the Online Exam.

1. Introduction

In this chapter, we will present the stages of implementation of our experimentation system and we present the results obtained. To do this, we will first present a brief overview of the language resources and the development environment used in our work, then the steps followed for the development of our system with Deep Learning as well as the tests carried out.

2. Part One: Architecture of the developed system

Our system monitors audiovisual cues of the test taker using a camera and a microphone. The cam is located above the monitor facing the test taker wish is referred to as the Webcam, the microphone is used to capture any sound cues in the room, we extract features from audio-visual streams utilizing five basic components(figure_8) while extracting those features the cheat engine estimate whether the test taker is cheating or not by following a set of rules pre-given then determine the possibility of cheating and then give's the result(figure_9) while taking records in some cases such as when a high sound is detected by the mic, or an unknown person is detected in the Webcam range of view. The system architecture is shown in figure_10.

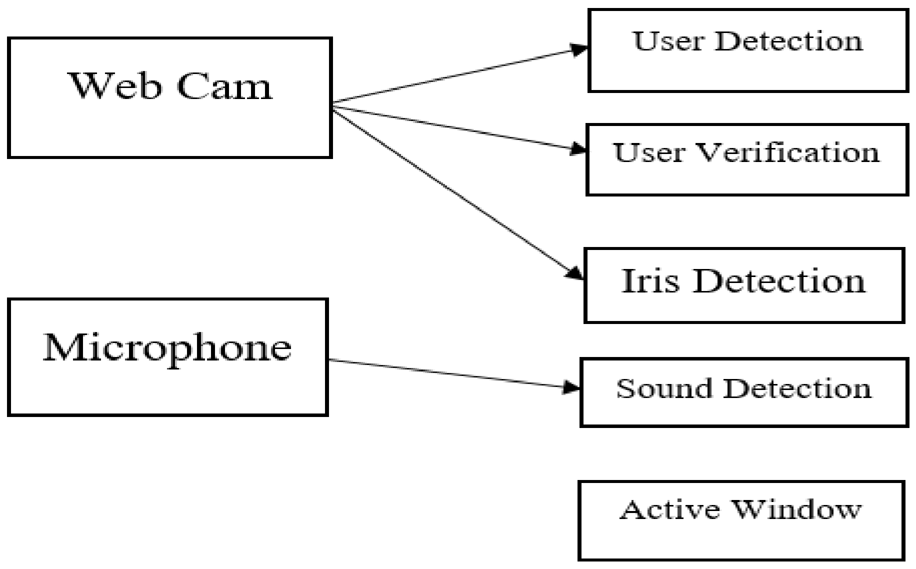


Figure 8 System features

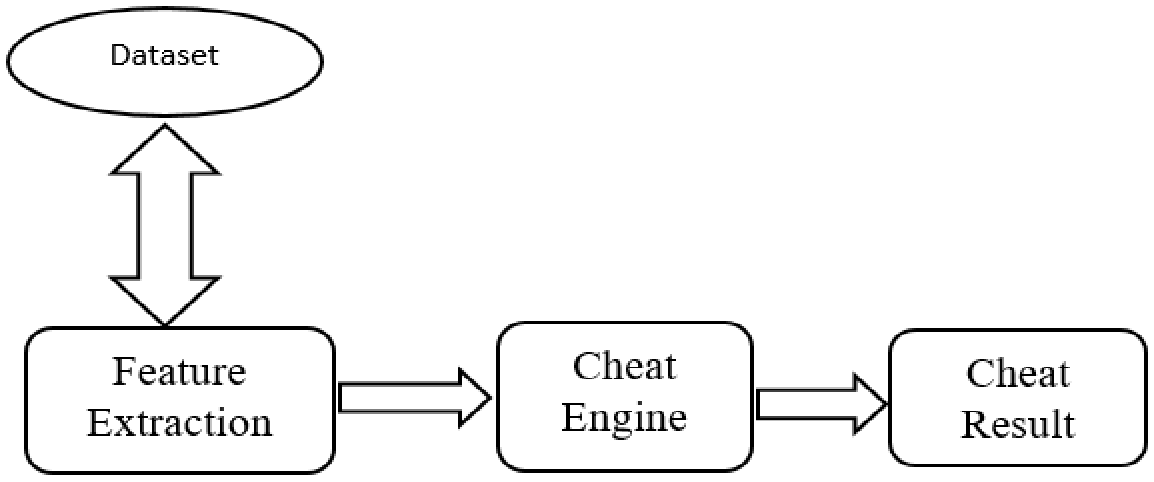


Figure 9 Result proses

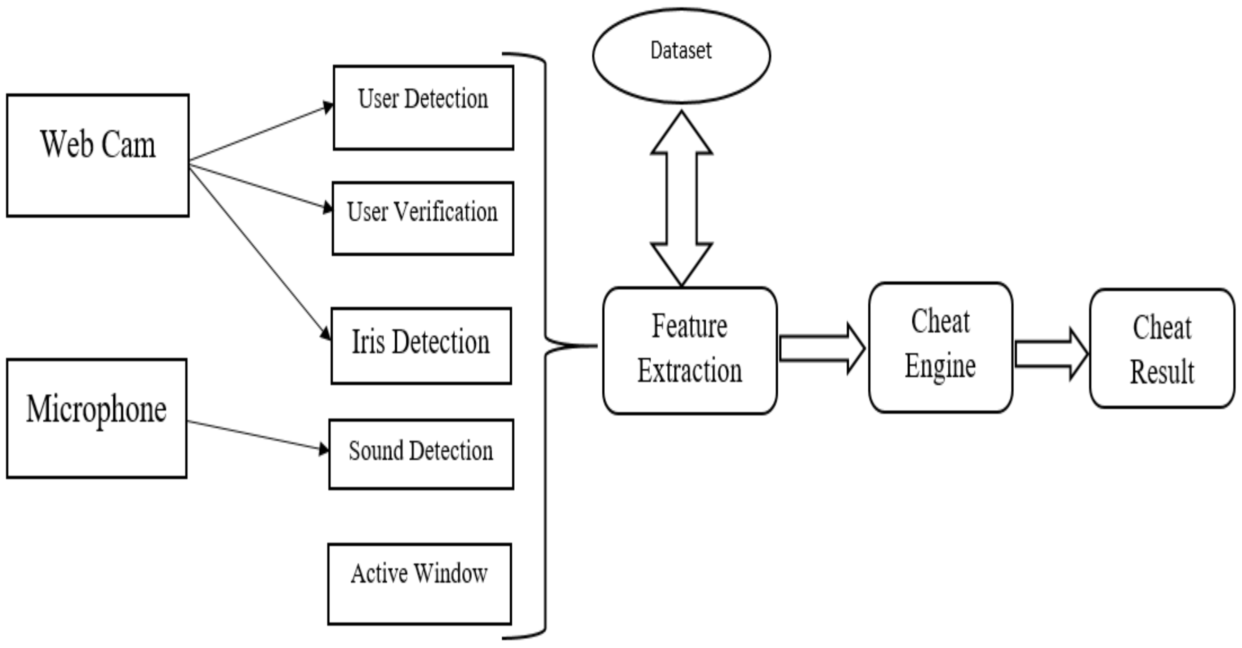


Figure 10 System Architecture

For user detection and verification, we use deep learning-based facial recognition including the concept of "deep metric learning". For this, we use Dlib facial recognition network (CNN). Our network architecture for face recognition is based on ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al., but with fewer layers and the number of filters reduced by half.

The network itself was trained by Davis King on a dataset of ~3 million images. On the Labeled Faces in the Wild (LFW) dataset, the network compares to other state-of-the-art methods, reaching 99.38% accuracy.

For sound detection, we will make use of the pyaudio package to detect high sound frequency and record it in some cases.

Our system will focus on face-recognition as we will first identify the test taker then follow his/her movements we also take note of the sound is the area while monitoring the active window. Therefore, if the test taker is not himself, he will be flagged or if there is too much movement or another person the frame with him, the system will also flag him if he makes too much noise or speak loudly.

To monitor the user movements, we will use the face location and coordinates while implementing a threshold on his movements and the sound frequency in his surroundings.

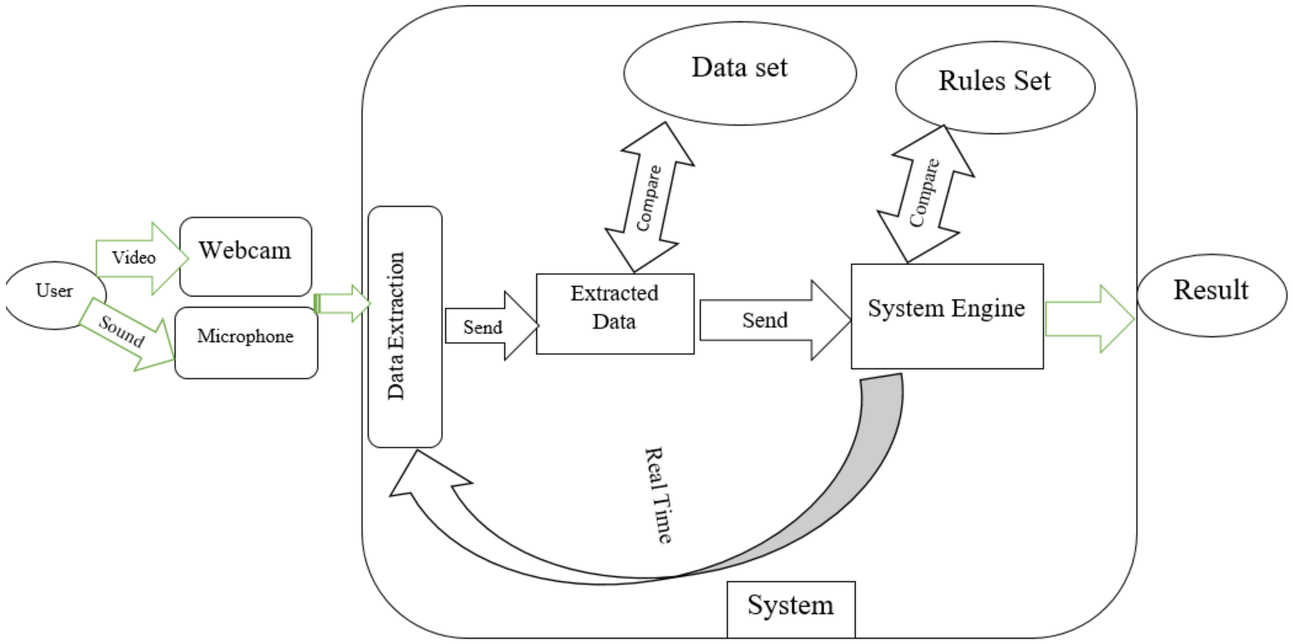


Figure 11 How the system work's

As shown in the figure above the user will be monitored in real time using the webcam and the mic, data will be extracted and compared to the data set we have and send them to the system engine if there is no problem with it. The engine will use a set of pre-given rules to make a decision on whether the test taker is cheating or not, all that in real time.

3. Part Two: The development tools environment

3.1. Equipment

The configuration of the hardware used in our implementation is as follows:

- Processor AMD R5 3400g with Radeon Vega graphics (4C/8T 4.2GHZ 6MB 65W AM4).
- RAM DDR4-3000 Gaming 8GB.
- Disk Maxtor SSD 500GB 2.5.
- 64-bit Ubuntu 20.04.2 LTS operating system.

3.2. Python

In this work we used the programming language Python which is a high-level interpreted programming language (there is no compilation step), interactive and object oriented with dynamic semantics.

Python is designed to be highly readable and it frequently uses English keywords while other languages use punctuation, and it has fewer syntactic constructs than other languages. It is in great demand by a large community of developers and programmers.

Created by Guido van Rossum and released in 1991, Python has a design philosophy that emphasizes code readability; including using large spaces, and has a large and comprehensive standard library. Its main characteristics are:

Python is an interpreted language; the instruction is processed at the time of execution by the interpreter. Therefore, the python program does not need to be compiled before it is executed. This is similar to PERL and PHP.

Python is interactive; it offers the possibility of using a Python command prompt and interacting directly with the interpreter to write the programs.

Python is object oriented; it supports the style or technique of Object-oriented programming that encapsulates code in objects. **[w2]**

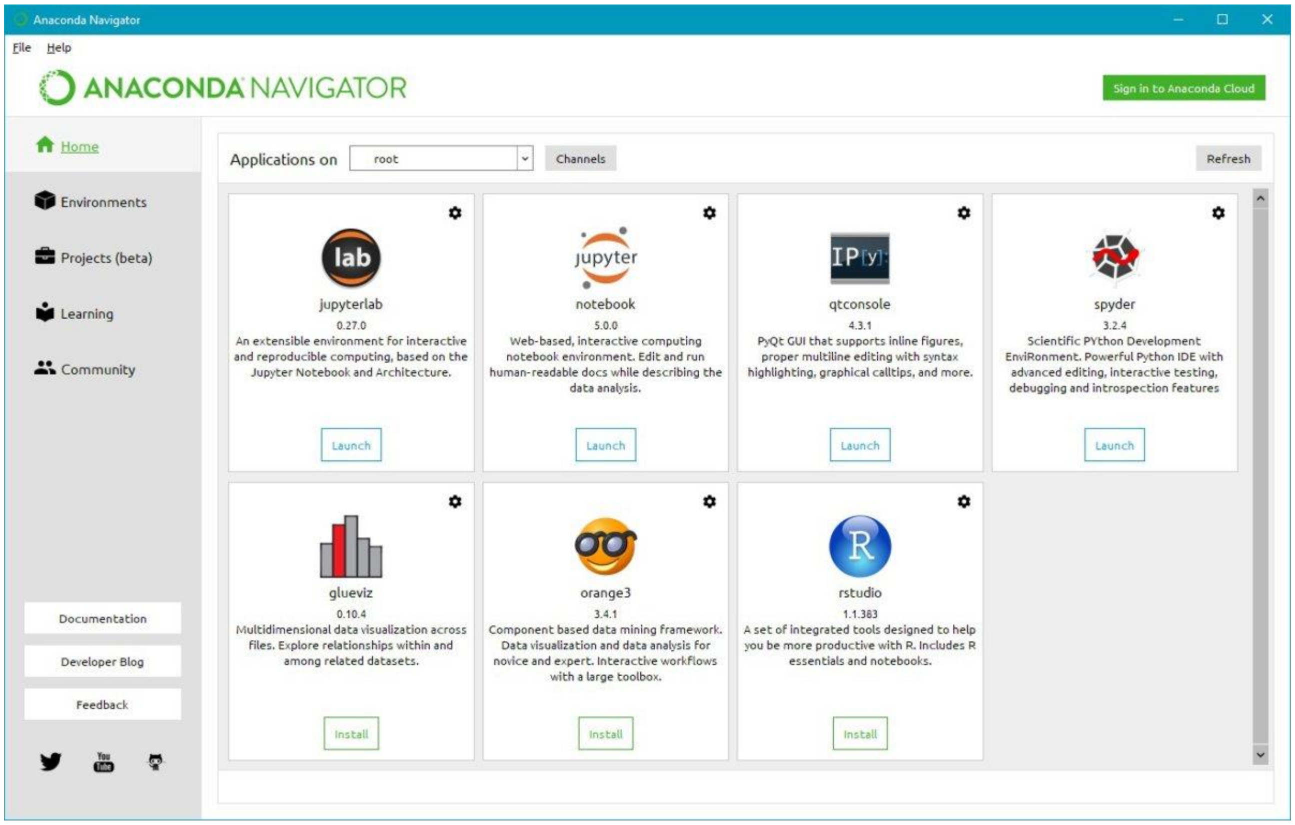
3.3. Spyder

Spyder is a free and open-source 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. **[w3]**

3.4. The Anaconda environments

Anaconda is a free, open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), which aims to simplify the management and development of packages (Package: set of programs designed to meet IT tasks). The distribution includes data science packages suitable for operating systems: Windows, Linux and macOS. It is developed and maintained by the "Anaconda, Inc." Company, which was founded by Peter Wang and Travis Oliphant in 2012.

Package versions in Anaconda are managed by the Conda Package Management System, this package manager was created as a separate open-source. There is also a scaled-down bootstrap version of Anaconda called Miniconda, which only includes Conda, Python, the packages they depend on, and a small number of other packages. Figure 8 presents the home window of the Anaconda environment. [W4]



jupyterlab
0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

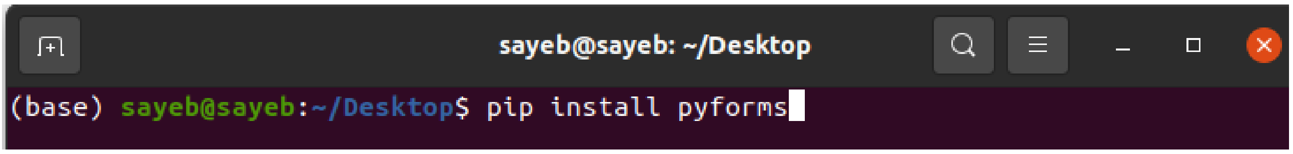
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

rstudio
1.1.383
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 12 The Anaconda environnements home page

4. Libraries used

To install any of the libraries bellow all you need to do is type this command in the terminal **Pip install '<u>library *name'*</u>** or follow the example in the figure bellow:-



```
(base) sayeb@sayeb:~/Desktop$ pip install pyforms
```

Figure 13 install library example

4.1. Numpy

Numpy is a library for performing numeric arithmetic operations in Python. The basic package around which the scientific computation stack is built is called Numpy (Numerical Python).

Numpy library provides easier handling of number tables and complex functions (propagation). It also provides an abundance of useful features for n-array and array operations in python, in addition to the guidance provided for Numpy array type math operations, which improves performance and thus speeds execution.

4.2. TensorFlow

TensorFlow is an open-source software library for high performance numerical computing. Created by the Google team in to facilitate the creation of machine learning modules. Its flexible architecture allows easy deployment of compute on various platforms (CPU, GPU, TPU), from desktops to server clusters. Even though TensorFlow was designed for neural networks, it works well for other networks where the computation can be modeled as a graph of data flow.

TensorFlow also uses several features of Theano such as common and subexpression elimination, automatic differentiation, shared and symbolic variables.

TensorFlow allows you to build several different types of deep networks, such as convolutional networks.

4.3. Keras

Keras is considered a powerful, easy-to-use Python library for developing and evaluating deep learning models. It has a minimalist design that allows you to build a network layer by layer; train him and run it. It encompasses the efficient Theano and TensorFlow numerical computational libraries and allows you to define and train neural network models in a few short lines of code.

It is a high-level neural network Application-programming interface (API), helping to widely use deep learning and artificial intelligence. It runs on top of a number of lower-level libraries including TensorFlow, Theano, etc. The Keras code is portable; we can implement neural network in Keras using Theano or TensorFlow as back end without any code change.

4.4. Cmake

Cmake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native make files and workspaces that can be used in the compiler environment of your choice.

The suite of Cmake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

4.5. Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge.

Major Features:

- Documentation
- High Quality Portable Code
- Deep Learning Algorithms

- Numerical Algorithms
- Graphical Model Inference Algorithms
- Image Processing

4.6. Face-recognition

The face recognition library, created by Adam Geitgey, wraps around Dlib's facial recognition functionality, making it easier to work with.

4.7. OpenCV-python

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing; video capture and analysis including features like face detection and object detection.

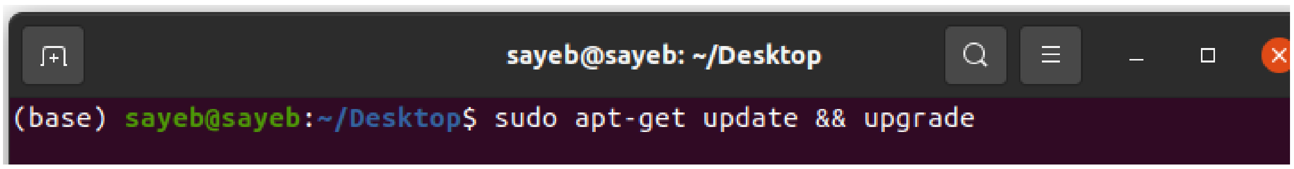
4.8. Pyaudio

Pyaudio provides Python bindings for Port Audio, the cross-platform audio I/O library. With Pyaudio, you can easily use Python to play and record audio on a variety of platforms. Pyaudio is inspired by:

1. pyPortAudio/fastaudio: Python bindings for PortAudio v18 API.
2. tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.

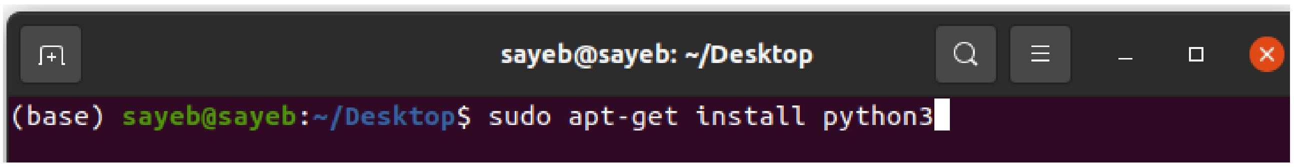
5. System and environment configuration

To start our work, we must first install and configure our operating system 'Ubuntu 20.04.2 LTS' install both python_3 and anaconda with the latest versions create our work environment and activating it, then we install all our packages, those steps are shown in the figures below:



```
sayeb@sayeb: ~/Desktop
(base) sayeb@sayeb:~/Desktop$ sudo apt-get update && upgrade
```

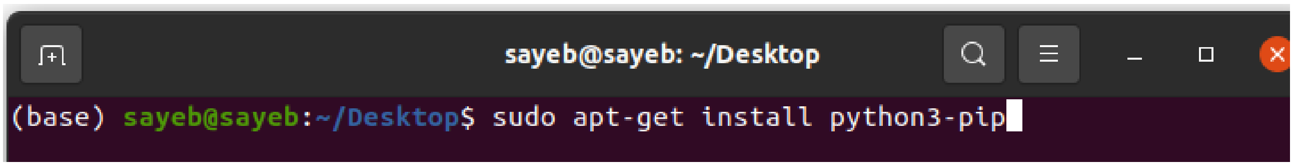
This figure below is to install python3:



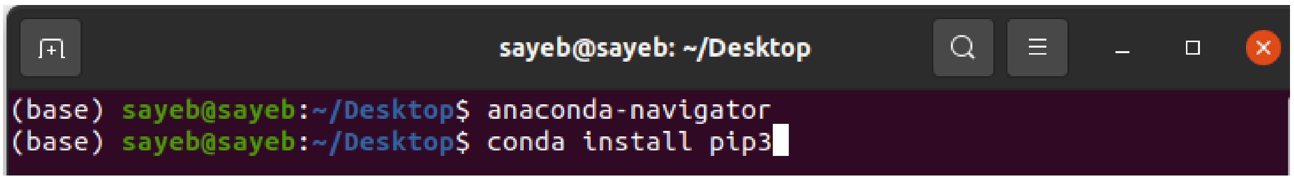
```
sayeb@sayeb: ~/Desktop
(base) sayeb@sayeb:~/Desktop$ sudo apt-get install python3
```

Figure 14 install python

The two figures below help facilitate future packages installment:



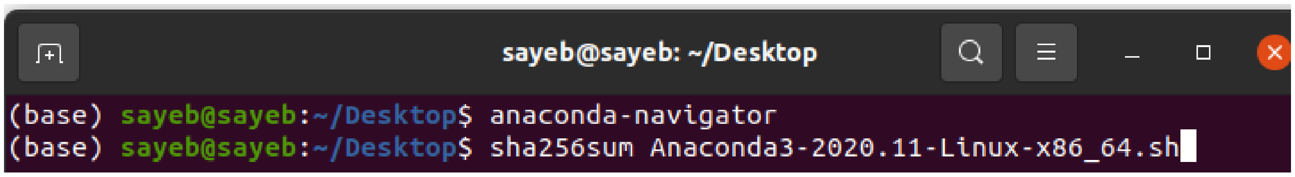
```
sayeb@sayeb: ~/Desktop
(base) sayeb@sayeb:~/Desktop$ sudo apt-get install python3-pip
```



```
sayeb@sayeb: ~/Desktop
(base) sayeb@sayeb:~/Desktop$ anaconda-navigator
(base) sayeb@sayeb:~/Desktop$ conda install pip3
```

Figure 15 install pip & pip3

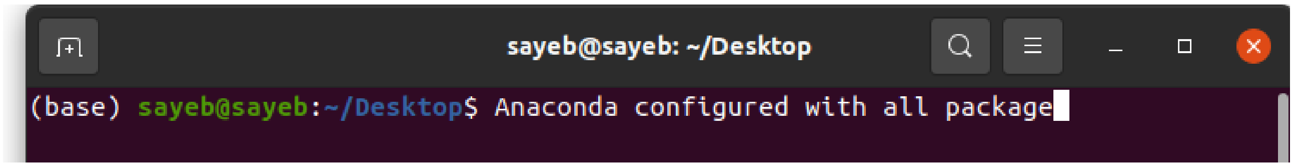
The figure below is to install anaconda wish we downloaded before:



```
sayeb@sayeb: ~/Desktop
(base) sayeb@sayeb:~/Desktop$ anaconda-navigator
(base) sayeb@sayeb:~/Desktop$ sha256sum Anaconda3-2020.11-Linux-x86_64.sh
```

Figure 16 install anaconda

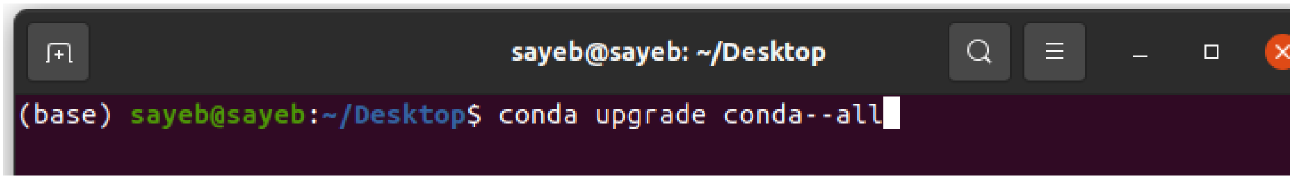
Here we configure anaconda with all packages:



```
(base) sayeb@sayeb:~/Desktop$ Anaconda configured with all package
```

Figure 17 configure conda packages

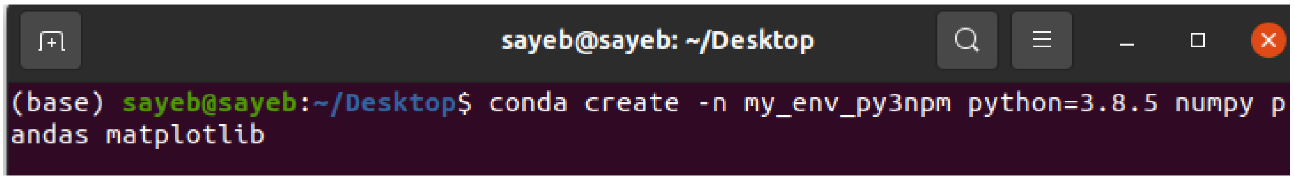
We upgrade conda to the latest version:



```
(base) sayeb@sayeb:~/Desktop$ conda upgrade conda--all
```

Figure 18 upgrade conda

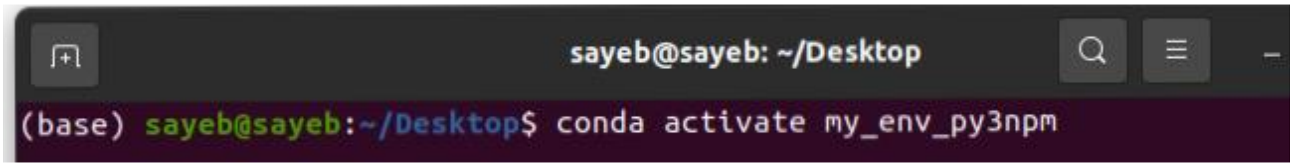
We create our conda environment by typing:



```
(base) sayeb@sayeb:~/Desktop$ conda create -n my_env_py3npm python=3.8.5 numpy p
andas matplotlib
```

Figure 19 create conda environment

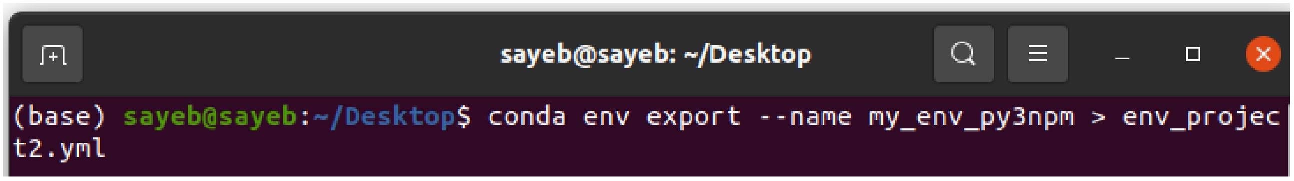
Then activate it:



```
(base) sayeb@sayeb:~/Desktop$ conda activate my_env_py3npm
```

Figure 20 activate conda environment

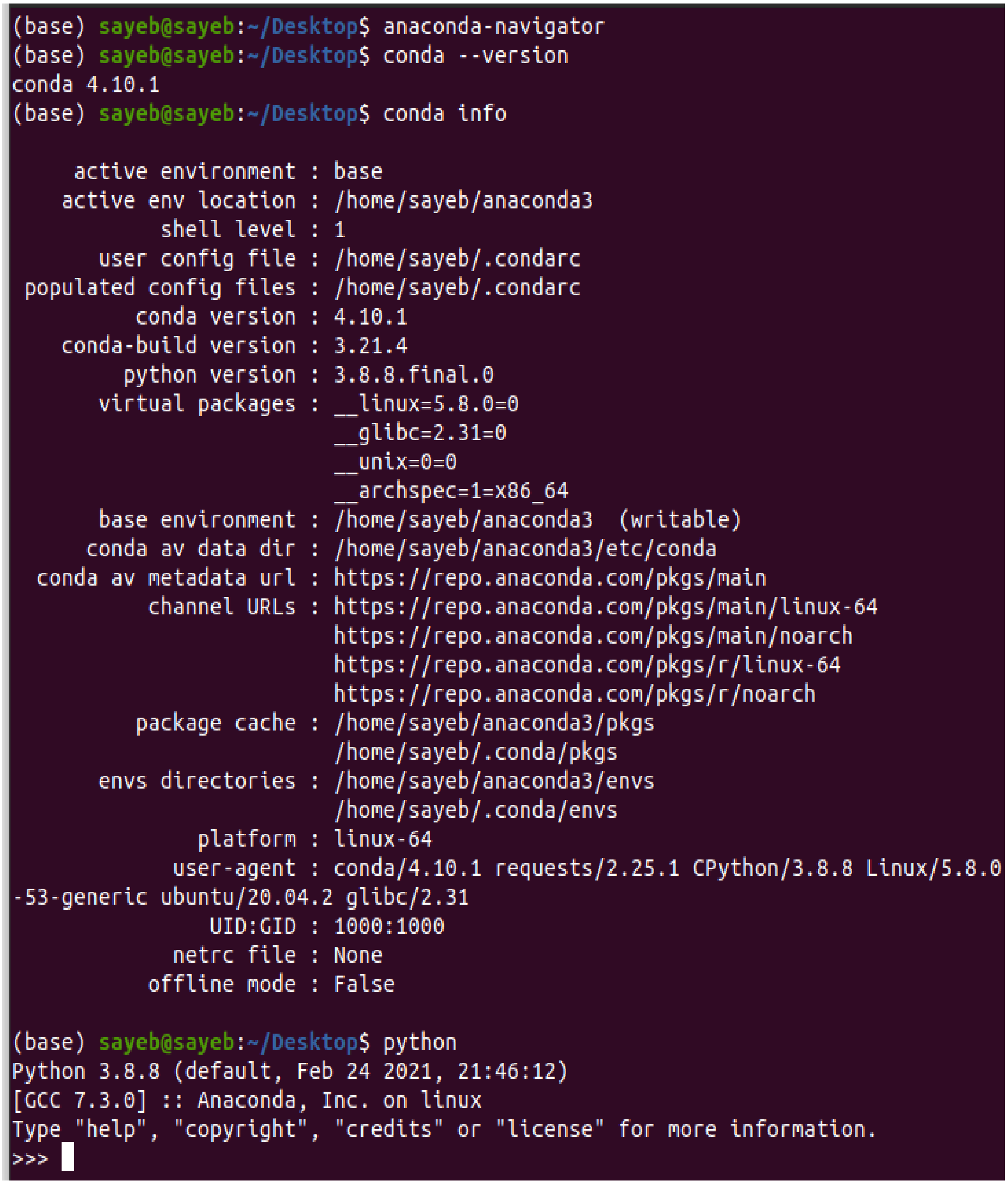
In addition, export it to save its progress:



```
(base) sayeb@sayeb:~/Desktop$ conda env export --name my_env_py3npm > env_projec
t2.yml
```

Figure 21 export conda environment

This figure shows our environment version and information's related to it:



```
(base) sayeb@sayeb:~/Desktop$ anaconda-navigator
(base) sayeb@sayeb:~/Desktop$ conda --version
conda 4.10.1
(base) sayeb@sayeb:~/Desktop$ conda info

     active environment : base
    active env location : /home/sayeb/anaconda3
            shell level : 1
       user config file : /home/sayeb/.condarc
 populated config files : /home/sayeb/.condarc
          conda version : 4.10.1
    conda-build version : 3.21.4
         python version : 3.8.8.final.0
       virtual packages : __linux=5.8.0=0
                          __glibc=2.31=0
                          __unix=0=0
                          __archspec=1=x86_64
       base environment : /home/sayeb/anaconda3  (writable)
      conda av data dir : /home/sayeb/anaconda3/etc/conda
  conda av metadata url : https://repo.anaconda.com/pkgs/main
           channel URLs : https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /home/sayeb/anaconda3/pkgs
                          /home/sayeb/.conda/pkgs
       envs directories : /home/sayeb/anaconda3/envs
                          /home/sayeb/.conda/envs
               platform : linux-64
             user-agent : conda/4.10.1 requests/2.25.1 CPython/3.8.8 Linux/5.8.0
-53-generic ubuntu/20.04.2 glibc/2.31
                UID:GID : 1000:1000
             netrc file : None
           offline mode : False

(base) sayeb@sayeb:~/Desktop$ python
Python 3.8.8 (default, Feb 24 2021, 21:46:12)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Figure 22 conda info's

6. Application

Here we will present our results; the two figures below present the cases of known and unknown faces:

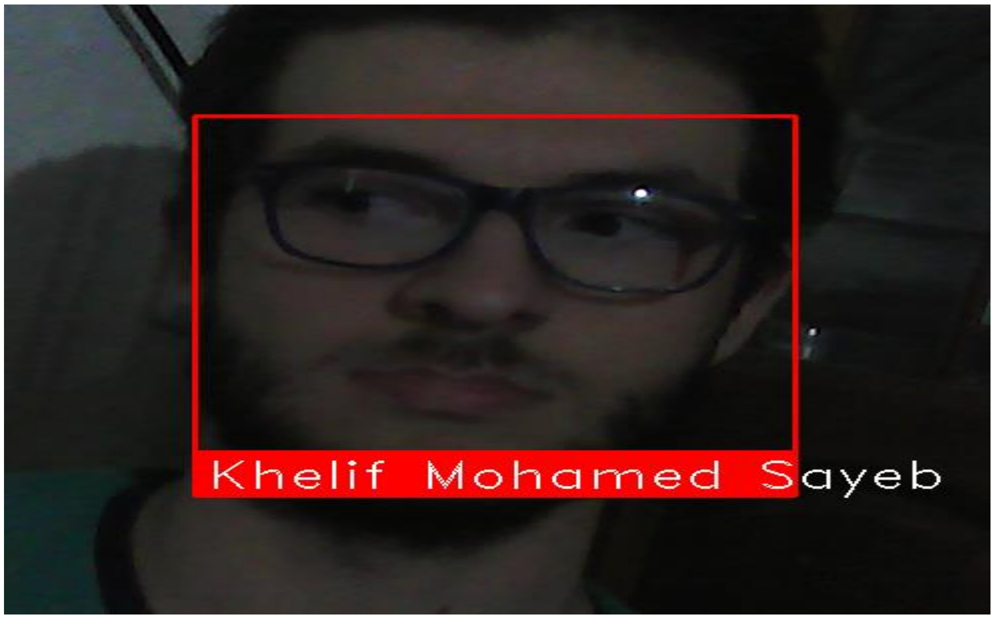


Figure 23 Example of a known face

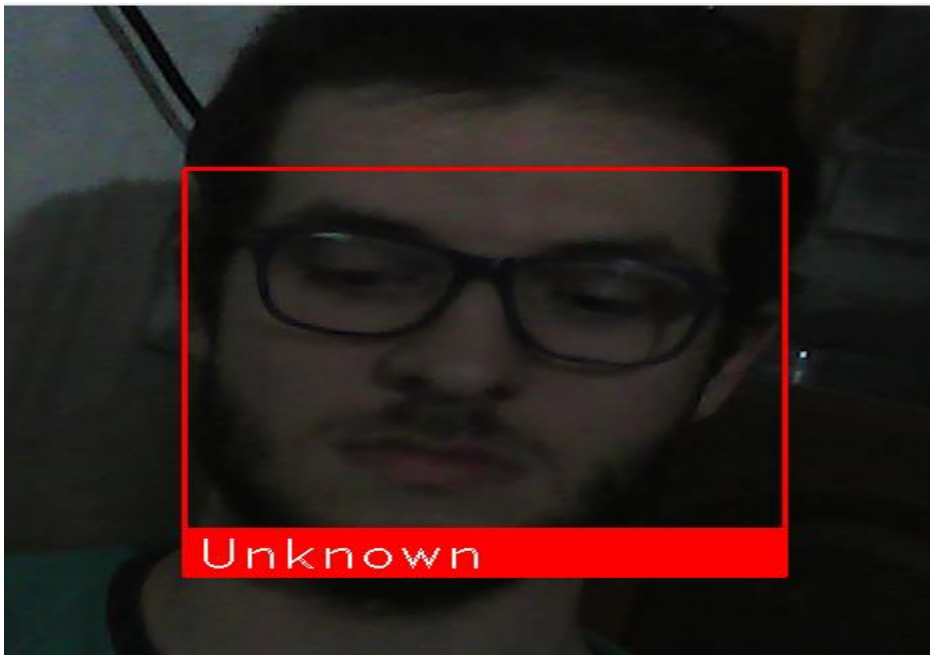
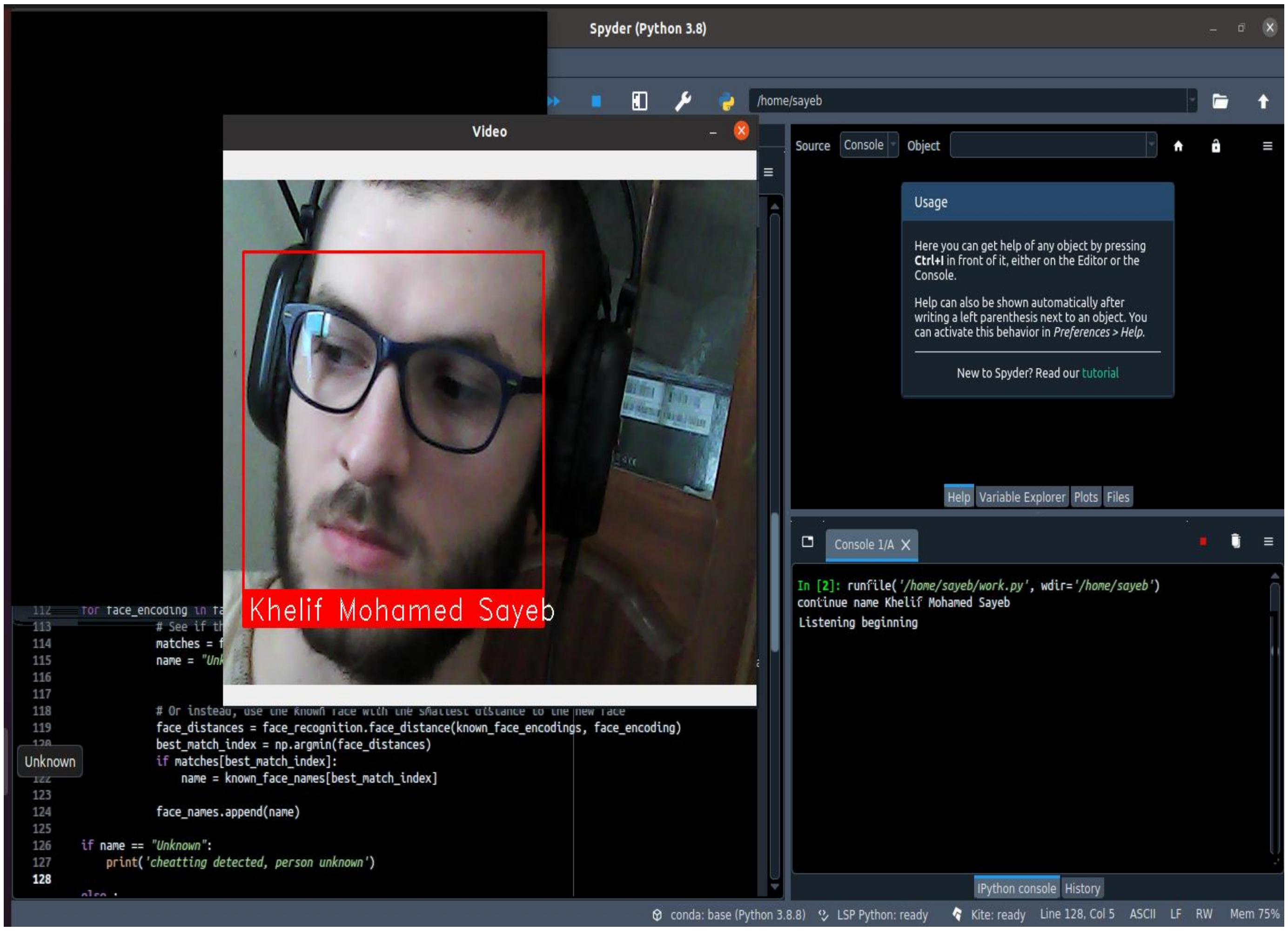


Figure 24 Example of an unknown face

The system starts by taking one frame and running face recognition on it if the person is unknown the system will not continue, but if the test taker exist in the data set the exam will continue and the listening will start

This figure here present the beginning of the test in case of a known test taker where his name is shown and the sound listening starts:



Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

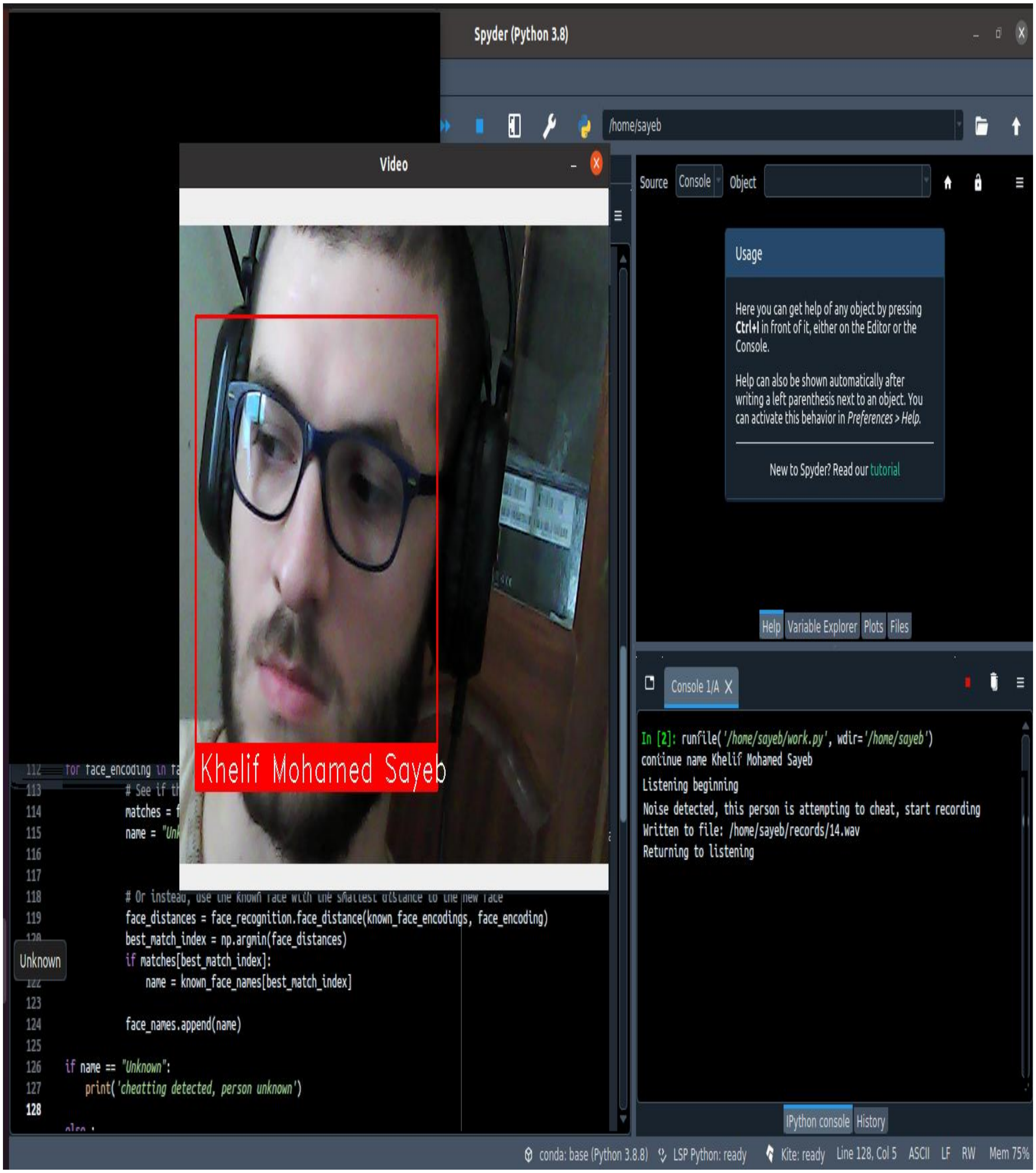
New to Spyder? Read our tutorial

```
        for face_encoding in fa
            # See if th
            matches = f
            name = "Unk


            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    if name == "Unknown":
        print('cheatting detected, person unknown')

```

```
In [2]: runfile('/home/sayeb/work.py', wdir='/home/sayeb')
continue name Khelif Mohamed Sayeb
Listening beginning
```

Figure 25 system starts after known face

In the next figure, a case is presented were sound is detected and the test taker is flagged with attempting to cheat and the recording starts to come back to the listening state a while later, storing the record for later use:



Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

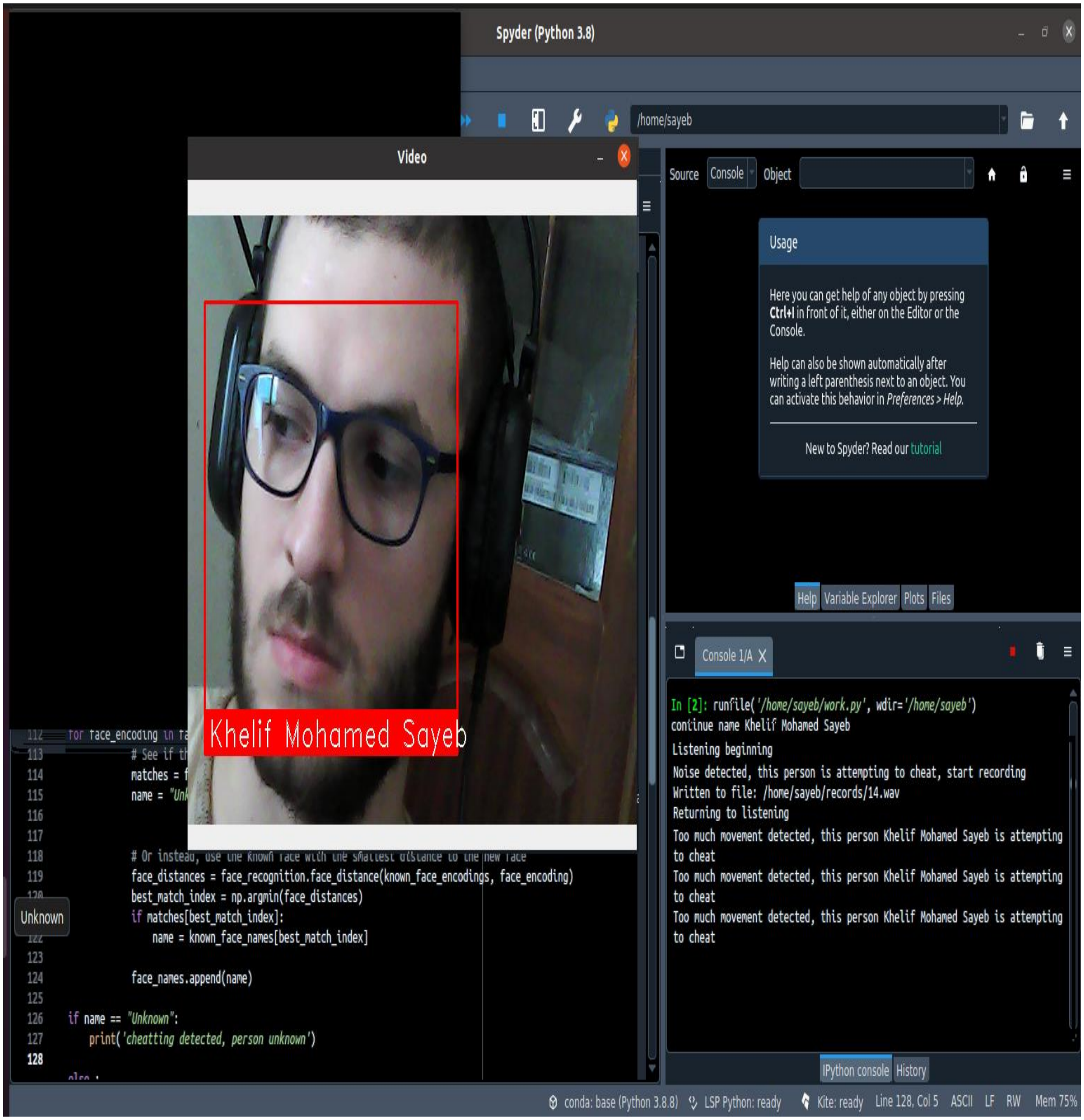
```
    for face_encoding in fa
            # See if th
            matches = f
            name = "Unk


            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    if name == "Unknown":
        print('cheatting detected, person unknown')

    else :
```

```
In [2]: runfile('/home/sayeb/work.py', wdir='/home/sayeb')
continue name Khelif Mohamed Sayeb
Listening beginning
Noise detected, this person is attempting to cheat, start recording
Written to file: /home/sayeb/records/14.wav
Returning to listening
```

conda: base (Python 3.8.8)   LSP Python: ready   Kite: ready   Line 128, Col 5   ASCII   LF   RW   Mem 75%

Figure 26 system detect sound and records it

While this figure 27 show the case of too much movements from the test taker wish exceeds the threshold that was set in this case the test taker is flagged with attempting cheat:



Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

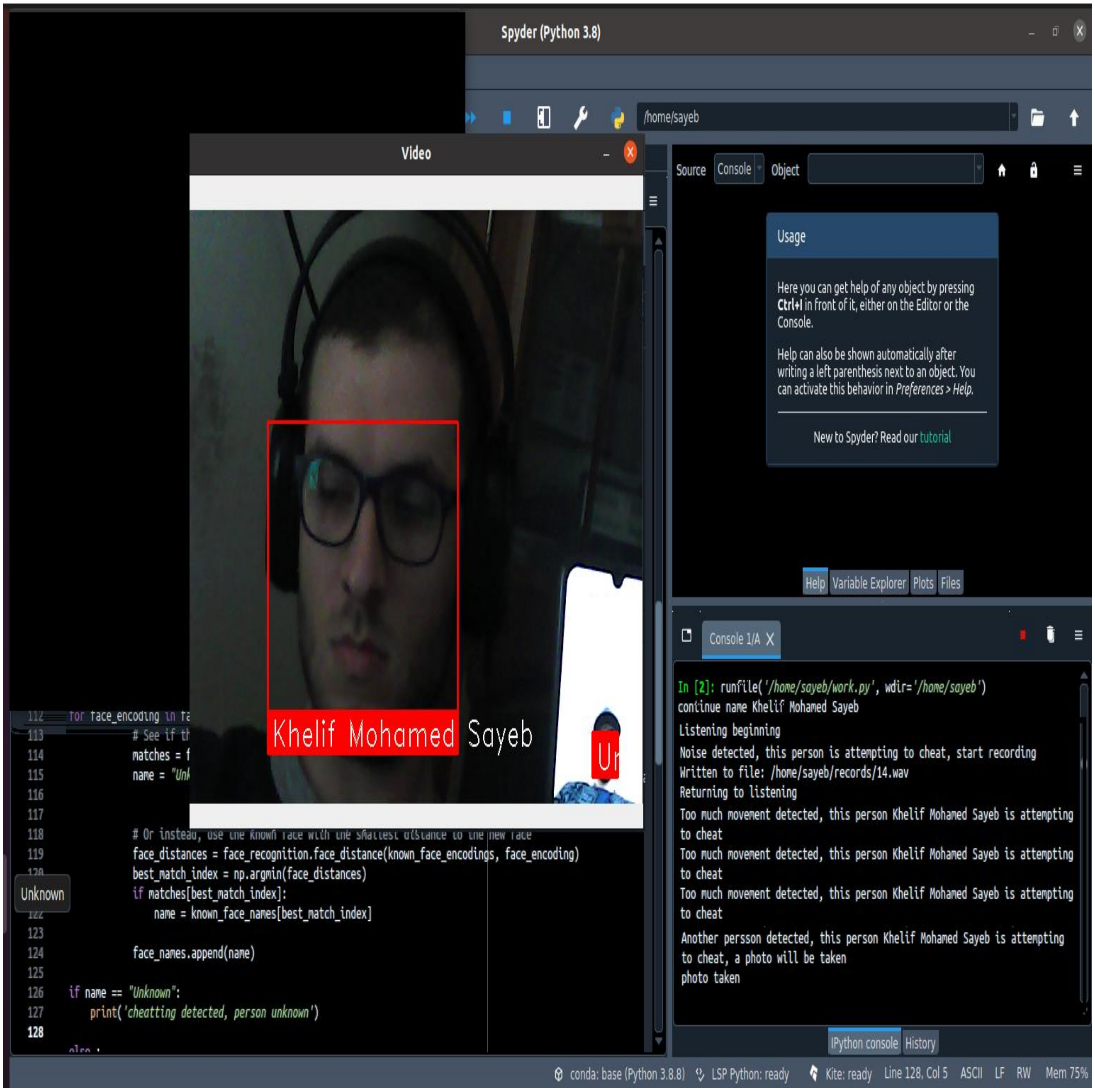
New to Spyder? Read our tutorial

```
    for face_encoding in fa
            # See if th
            matches = f
            name = "Unk


            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    if name == "Unknown":
        print('cheatting detected, person unknown')

    else :
```

```
In [2]: runfile('/home/sayeb/work.py', wdir='/home/sayeb')
continue name Khelif Mohamed Sayeb
Listening beginning
Noise detected, this person is attempting to cheat, start recording
Written to file: /home/sayeb/records/14.wav
Returning to listening
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
```

Figure 27 system detect to much movements

In the next figure 28, the case of another person appearing next to the test taker; in this case the system notifies us that the test taker is attempting to cheat, and also take a photo and store's it for later use



Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
    for face_encoding in fa
            # See if th
            matches = f
            name = "Unk


            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    if name == "Unknown":
        print('cheatting detected, person unknown')

    else :
```

```
In [2]: runfile('/home/sayeb/work.py', wdir='/home/sayeb')
continue name Khelif Mohamed Sayeb
Listening beginning
Noise detected, this person is attempting to cheat, start recording
Written to file: /home/sayeb/records/14.wav
Returning to listening
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
Too much movement detected, this person Khelif Mohamed Sayeb is attempting
to cheat
Another persson detected, this person Khelif Mohamed Sayeb is attempting
to cheat, a photo will be taken
photo taken
```

Figure 28 system detect another person

7. Conclusion

In this chapter we have presented the design of our system “a smart system to detect cheating in the online exam” wish allows the user to take the exam using simply his home computer without the presence of a supervisor to monitor him/her.

Our system monitors audiovisual cues of the test taker using a camera and a microphone. The cam is located above the monitor facing the test taker wish is referred to as the Webcam, the microphone is used to capture any sound cues in the room, we extract features from audio-visual streams utilizing five basic components while extracting those features the cheat engine estimate whether the test taker is cheating or not by following a set of rules pre-given then determine the possibility of cheating given us a result while taking records in some cases such as when a high sound is detected by the mic, or an unknown person is detected in the Webcam range of view.

The system can be further upgraded by integrating other AI features.

We would like to point that although our system is automated and offer a solution to problem it does not replace the human being and can never do it, it simply aids in this exceptional crisis.

General Conclusion & Perspectives

Conclusion

This year 2021 is considered an exceptional year because it has known the hardest health and economic crisis history due to the spread of the Coronavirus pandemic, wish led the whole globe to follow some serious downfall in terms of economy, even the day-to-day activities are being restricted due to enforcements like lockdown. However, with the blessings of technology, the strings of knowledge and learners are still intact. However, not the same can be said for exams.

This is where our work provides a solution. Our system, which is a smart system to detect cheating in the online exam, allows taking the exam using only your computer avoiding making contact with other people. Our work, make use of AI and Deep Learning technology's such as face recognition, face detection, sound detection and active window detection to deliver a solution and help conduct exams in a controlled environment.

Our system monitors audiovisual cues of the test taker using a camera and a microphone. The cam is located above the monitor facing the test taker wish is referred to as the Webcam, the microphone is used to capture any sound cues in the room, we extract features from audio-visual streams utilizing five basic components while extracting those features the cheat engine estimate whether the test taker is cheating or not by following a set of rules pre-given then determine the possibility of cheating given us a result while taking records in some cases such as when a high sound is detected by the mic, or an unknown person is detected in the Webcam range of view.

Perspectives

This work can be exploited for several diverse applications, can be added to or reshaped to serve other purposes. This work although it helps it is not the best solution as it does not replace the human monitoring but offers aid.

In the end, this work was very beneficial for us because it allowed us to learn many new concepts that we did not learn during our course of training, such as AI, deep learning and intelligent techniques, programming with python software, etc. We hope this work helps future students wishing to work on this subject.

References

Bibliography

[1] CHOLLET Francois,Deep learning with python, November 2017.

[2] Boughaba Mohammed et Boukhris Brahim., L`apprentissage profond (Deep Learning) pour la classification et la recherche d'images par le contenu. Thèse de master: University Kasdi Merbah, Ouargla .2016-2017.

[3] Maria ZIMINA, l'apprentissage automatique appliqué à l'évaluation de la traduction automatique, UNIVERSITÉ PARIS DIDEROT (PARIS 7).

[4] Marref Nadia, Apprentissage Incrémental and Machine Vecteurs Supports, Thèse de master : Université Batna.

[5] Julien Ah-Pine, Université Lyon 2M DM 2019/2020.

[6] Hassane Hilali, Application De La Classification Textuelle Pour L'extraction Des Règles D'association Maximales, avril 2009.

[7] Youcef Djeriri, Les Réseaux de Neurones Artificiels, septembre 2017.

[8] Labiad Ali, Sélection Des Mots Clés Basée Sur La Classification Et L'extraction Des Règles D'association, Juin 2017.

[9] ZACCONE Giancarlo, MD REZAUL Karim, MENSHAWY Ahmed. Deep learning with tensorflow 2017.

[10] DIALLO Nene Adama Dian, La reconnaissance de l'expressions faciales ; Thèse de master : Université 8 Mai 1945, Guelma. Juillet 2019.

[11] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104(2), 154-171.

[12] PARIZEAU Marc ; Réseaux de neurones ; 2004.

[13] J.F. Jodouin, "Les réseaux de neurones. Principes et définitions", édition Hermis, Paris, 1994.

[14] Trask, A. W. (2019). Deep learning.

[15] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature.

[16] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and trends in signal processing.

[17] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision.

[18] Charu C. Aggarwal, Neural Networks and Deep Learning, IBM T. J. Watson Research Center International Business Machines,Yorktown Heights, NY, USA.

[19] Phil Kim; Matleb Deep learning with machine learning, neural networks and Artificial intelligence.

[20] Li, Stan Z.; Jain, Anil K. (2005). Handbook of Face Recognition. Springer Science & Business Media.

[21] de Leeuw, Karl; Bergstra, Jan (2007). The History of Information Security: A Comprehensive Handbook. Elsevier.

[w1] : https://fr.mathworks.com/discovery/deep-learning.html .

[w1] : https://www.saagie.com/fr/blog/qu-est-ce-que-le-deep-learning

[w2] https://www.python.org/

[w3] https://koor.fr/Python/Tutorial/python_ide_pycharm.wp

[w4] https://jupyter.org/

[w5] https://en.wikipedia.org/wiki/Anaconda_ (Python_distribution)

Annex

Annex

This here is the part of the code responsible for detecting sound and recording it: -

```
import pyaudio
import math
import struct
import wave
import time
import os

Threshold = 10
SHORT_NORMALIZE = (1.0/32768.0)
chunk = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 16000
swidth = 2
TIMEOUT_LENGTH = 5
f_name_directory = r'/home/sayeb/records'

class Recorder:
    @staticmethod
    def rms(frame):
        count = len(frame) / swidth
        format = "%dh" % (count)
        shorts = struct.unpack(format, frame)
        sum_squares = 0.0
    for sample in shorts:
        n = sample * SHORT_NORMALIZE
        sum_squares += n * n
    rms = math.pow(sum_squares / count, 0.5)
    return rms * 1000
def __init__(self):
    self.p = pyaudio.PyAudio()
    self.stream = self.p.open(format=FORMAT,
                              channels=CHANNELS,
                              rate=RATE,
                              input=True,
                              output=True,
                              frames_per_buffer=chunk)
def record(self):
    print('Noise detected, this person is attempting to cheat, start recording')
    rec = []
    current = time.time()
    end = time.time() + TIMEOUT_LENGTH
    while current <= end:
        data = self.stream.read(chunk)
        if self.rms(data) >= Threshold: end = time.time() + TIMEOUT_LENGTH
        current = time.time()
        rec.append(data)
    self.write(b''.join(rec))
def write(self, recording):
    n_files = len(os.listdir(f_name_directory))
    filename = os.path.join(f_name_directory, '{}.wav'.format(n_files))
    wf = wave.open(filename, 'wb')
```

```
        wf.setnchannels(CHANNELS)
        wf.setsampwidth(self.p.get_sample_size(FORMAT))
        wf.setframerate(RATE)
        wf.writeframes(recording)
        wf.close()
        print('Written to file: {}'.format(filename))
        print('Returning to listening')
    def listen(self):
        print('Listening beginning')
        while True:
            input = self.stream.read(chunk)
            rms_val = self.rms(input)
            if rms_val > Threshold:
                self.record()
```

And this down here is the part for detecting and recognizing the user's face: -

```
import face_recognition
import cv2
import numpy as np

video_capture = cv2.VideoCapture(0)
sayebb_image = face_recognition.load_image_file("/home/sayeb/images/sayebb.jpg")
sayebb_face_encoding = face_recognition.face_encodings(sayebb_image)[0]
known_face_encodings = [
    sayebb_face_encoding
]
```

```
known_face_names = [
    "Khelif Mohamed Sayeb"
]
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()
    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]
    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
        face_names = []
        for face_encoding in face_encodings:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown"

            # Or instead, use the known face with the smallest distance to the new face
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]
            face_names.append(name)
    process_this_frame = not process_this_frame
    # Display the results
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw the box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
    # Display the resulting image
    cv2.imshow('Video', frame)
```