5.8 CHEATING DETECTION SYSTEM

5.8.1 INTRODUCTION

The supervision of students in an exam is a manual time-consuming job that can be automated completely by using various machine learning algorithms in this section ML/DL techniques used in cheating detection process is highlighted and identified.

Creation of automated exam supervision require multiple parts to be figured out to mimic the real/original process of supervision:

1 – The student should be supervised not to have any tool to cheat with, the cheating objects can be divided into:

- Physical cheating tools like (Books, Mobile, screens, etc.)
    - For this purpose, object detection algorithm should be made to detect if a student is. Using any physical objects to cheat
- Voice-based cheating tools like (Someone in the room trying to help the student or any voice-based material that might be considered as a cheating tool)
    - For this purpose, a sound classifier must be able to detect if any student if cheating using voice-based ways.

2 – The student should not be able to look around him but only to look at his exam

- For this purpose, a face pose estimator must be implemented to detect if student is not looking straight into his exam for long periods of time

The diagram below summarizes the different models used in the cheating detection process

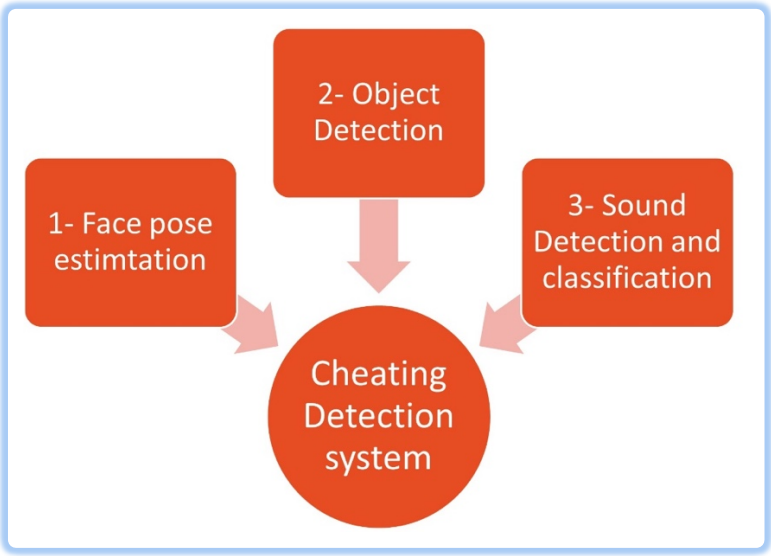


Figure 1: Cheating Detection Process Summary

5.8.2 FACE POSE ESTIMATION MODEL

Human face can be thought of a 3-D object that have three rotating axes which are known as Euler angles:

1- Pitch: It identifies the rotation of the face in terms of up and down it can be thought of as Y-axis rotation of face
2- Yaw: it identifies the rotation of face in terms of right of left it can be thought of as the x-axis of the face
3- Roll : it identifies the z-axis of human face

Our objective which is detecting where are the students' face looking, at every time step in an exam, we are more concerned with the x-axis and y-axis movement which is yaw and pitch while neglecting the roll values

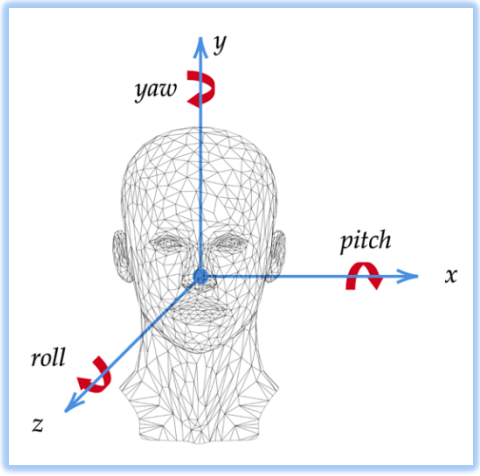


Figure 2: Pose Estimation

5.8.3 MODEL ARCHITECTURE

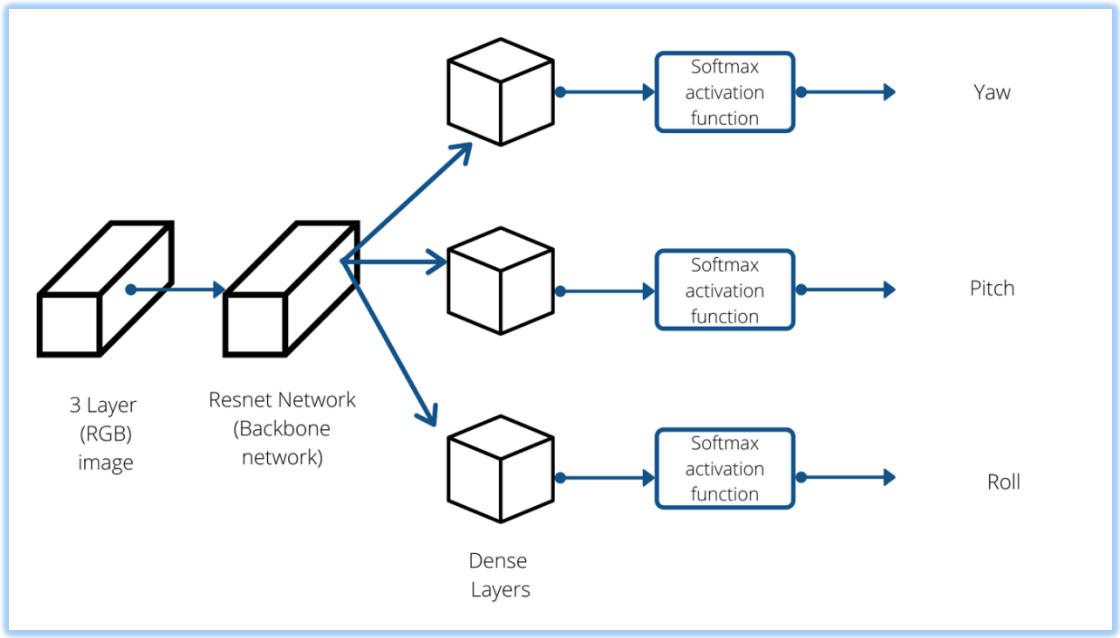


Figure 3: Face Pose Estimation Model Architecture

The Model Consists of 3 Main parts:

- The input part: the input part in which a colored image (RGB -Red Green Blue- image) is entered to the model.
- The second part: It contains the Backbone network which is residual neural network (Deep learning Convolutional Neural Network), Resnet has proven to be one of the best CNNs by introducing ideas like shortcuts (skip connections).

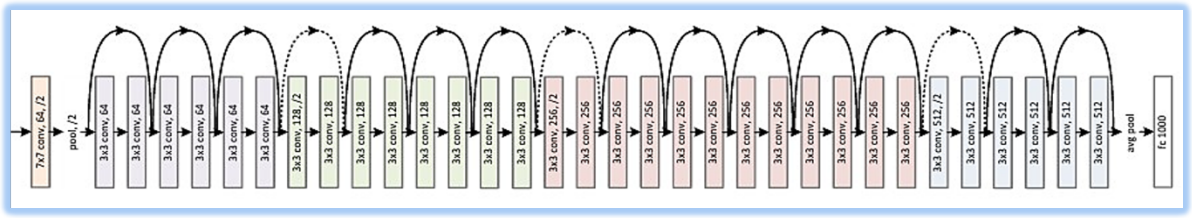


Figure 4: ResNet Model Diagram

The third and final part: contains three fully connected layers (Dense layers) Which is used to predict the yaw, pitch, and roll, all of the 3 connected layers have the output of previous Resnet as input.

5.8.4 MODEL LOSS FUNCTION

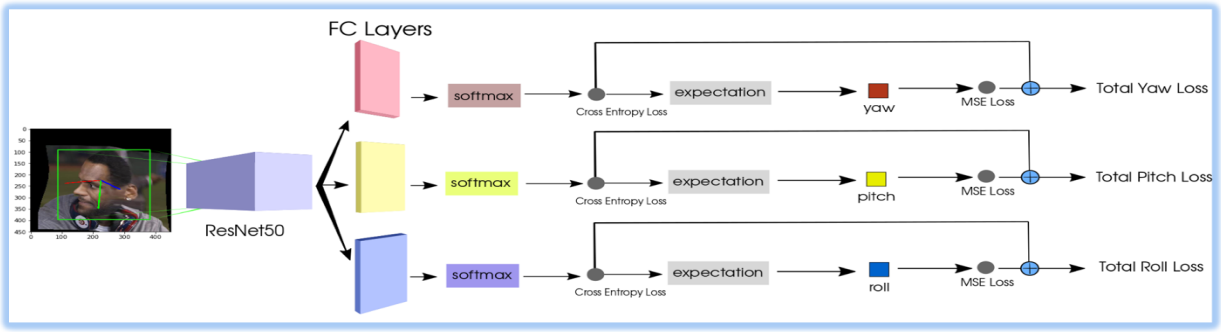


Figure 5: Face Pose Estimation Model

Model has three separate losses to be computed, one loss function per each angle; to have three different values to act upon and backpropagate into the network doing this accelerate and improves learning.
Each loss is a combination of two components:

$$\boldsymbol{L = H(y, \hat{y}) + \alpha \times mse(y, \hat{y})}$$

- Binned pose classification: Is a combination of softmax activation function and cross-entropy, thus the network learns to predict the neighbourhood of the pose in a robust fashion.
- Mean Square Error (MSE) regression loss function: in order to improve fine-grained predictions

5.8.5 PRE-MODEL SETUP

To make it easier and more accurate to predict the pose in the image some preprocessing should be done on the image.
The pre-processing steps is:

1- Read the original image (Using opencv).
2- Detect Human face position in the Image (using Dlib).
3- From step (2) define face position and add some margins around the face.
4- Crop the image by using step (3).

Then the image is ready to be processed by the model.

5.8.6 SIMPLIFIED SEQUENCE DIAGRAM FOR MODEL

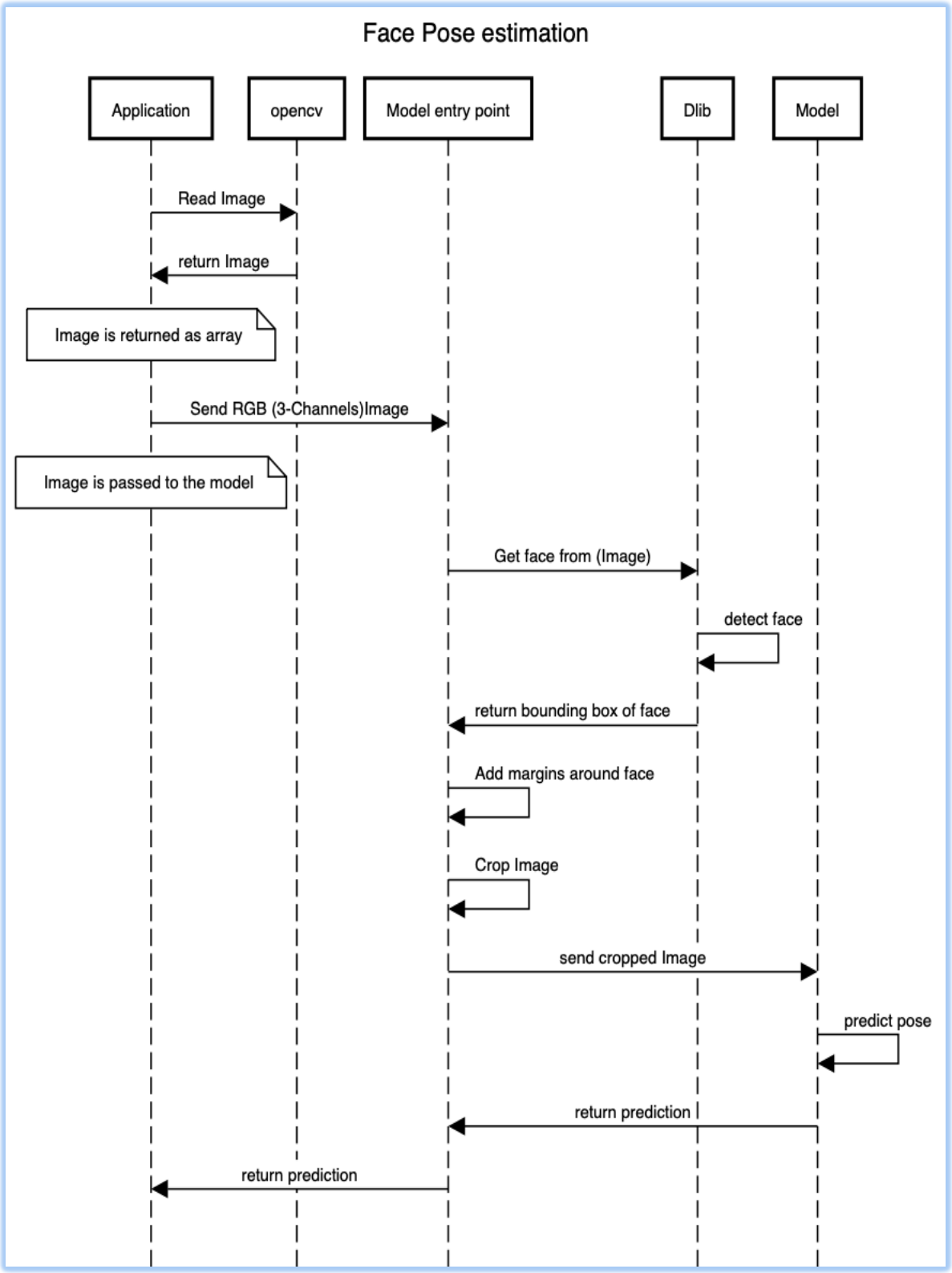


Figure 6: Sequence Diagram for Face Pose Estimation

5.8.7 SCORING SYSTEM FOR POSE ESTIMATION DETECTION

The scoring system of pose estimation can be considered as the post-processing step of head pose estimation.

The following steps summarize the scoring algorithm:

1) For every Image calculated pose in degrees transfer it to direction based pose
   a) For example instead of (0,0) to Center, Center
2) Create four arrays (lists) two for the calculated direction based poses of all images one for Horizontal axis and one for the Vertical axis (Yaw, Pitch) and the other two for the scores of the two poses lists
3) For Every lists of the two list iterate through its elements from second element till the end
   a) If current element is Center set element score as 0
   b) If current element is not Center and the previous element is Center set element score as 1
   c) If current element is not Center and the previous element is same direction (for example current direction is Up and previous is also Up) set element score as 1
   d) If current element is not Center and the previous element is opposite direction (for example current direction is Up and previous is also Down) set element score as 2
4) Sum the two scoring lists to get the final score
5) Divide the score / num of pictures to get the score rate
6) Compare the rate to a given threshold if it is beyond the given threshold then this considered as cheating case

5.8.7.1 SCORING SYSTEM FOR POSE ESTIMATION DETECTION EXAMPLE

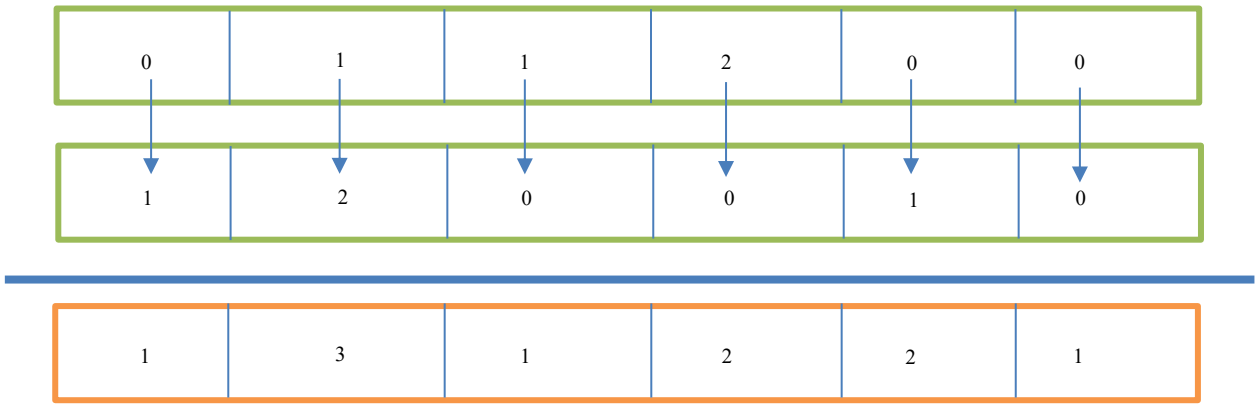
1. Arrays in red is direction-based pose arrays
2. Arrays in green is score arrays for each of the direction-based pose arrays
3. Array in orange is summation of score arrays

Direction Based Poses and their elements score

Center	Up	Up	Down	Center	Center
0	1	1	2	0	0

Left	Right	Center	Center	Left	Center
1	2	0	0	1	0

Calculation of overall score array



Calculation of Cheating Scores:

Score = 1+3+1+2+2+1 = 10 Score rate = 10/6 = 1.67

5.8.8 OBJECT DETECION MODEL

The objective of this model is to detect objects that can be suspected to be a cheating tool in the hands of the student, detecting these objects serve as a way to know if a student is trying to cheat.
An Example of objects that can be detected is illustrated in the diagram.

Yolo Framework is used as our object detection model

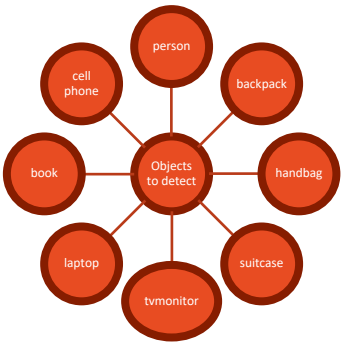


Figure 7: Example of Cheating Objects

5.8.9 HOW IT WORKS

You Only Look Once also known as YOLO is an object detection algorithm -Model- that has the ability not only to detect multiple objects from one scene at only one forward feed step to the network, but it also supports creating boxes indicating the location of each single object detected.

1. The main idea of the algorithm is that it combines the usage of Convolutional neural network with dividing the input's image into a grid (usually 13X13) of cells where:

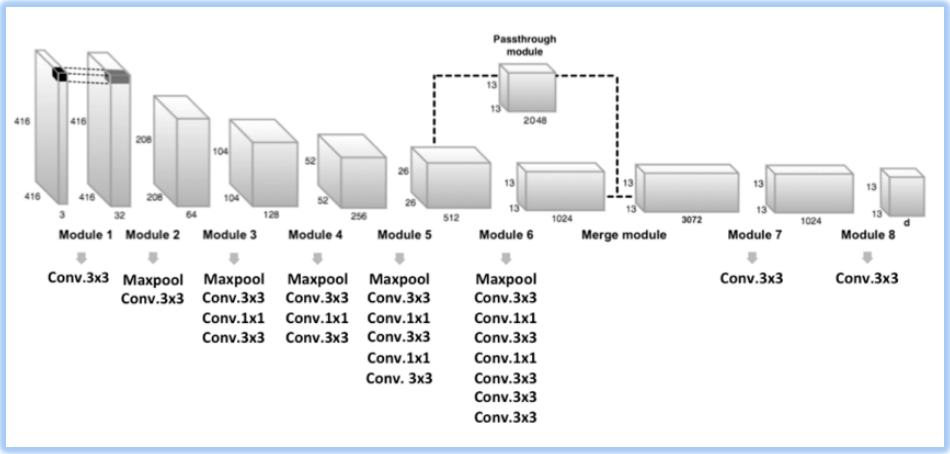


Figure 8: YOLO Model Architecture

- Every single cell creates five rectangles enclosing the detected object -Also known as bounding boxes-.
- Having multiple cells, the YOLO model uses them to predict the probability of having an object in a cell
  - Bounding Boxes is being weighted by the probability of a prediction

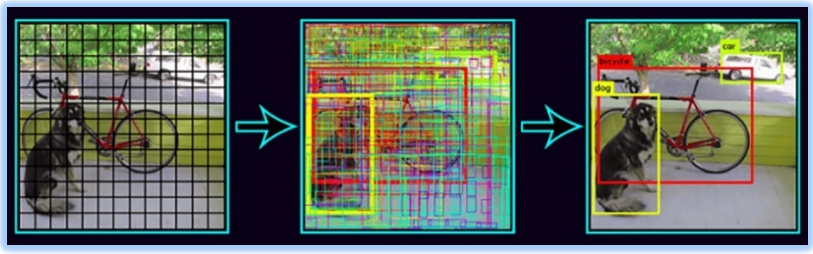


Figure 9: YOLO Object Detection Example

5.8.10 SOUND DETECTION MODEL

The objective of sound detection model is to serve as a detector for sounds around the student to ensure that he is taking his exam alone and doesn't have any human voice nearby.

To be able to deal with audio which can be considered as analog signal (Analog is defined as a signal that has a continuously and smoothly varying amplitude or frequency, Human speech, and nearly everything else humans hear, is in analog form.) we first must transform it into a digital form.

5.8.11 PULSE-CODE MODULATION

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest value within a range of digital steps.

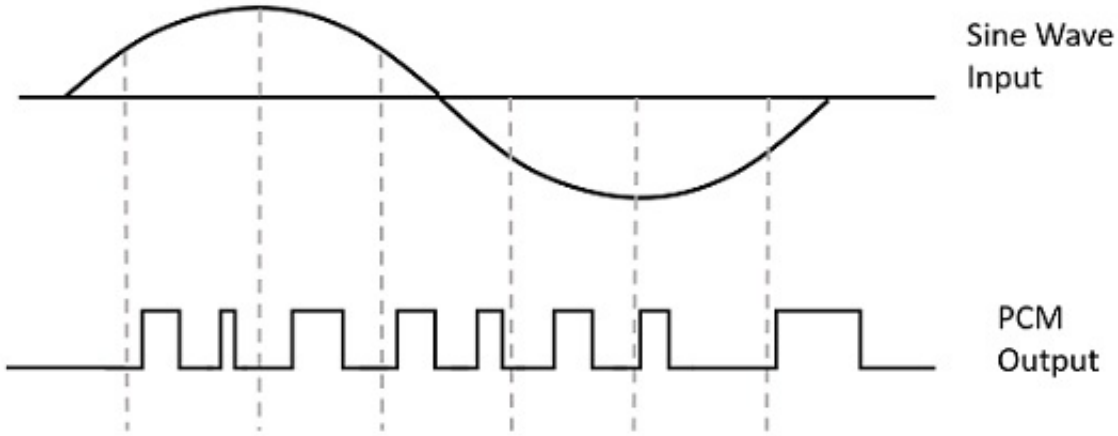


Figure 10 - PCM example

5.8.12 PRE-PROCESSING FOR AUDIO

1. Read original .wav file
2. Set Frame rate to 48,000 (Down sampling -in some cases up sampling -)
3. Set Channels to mono (1)
4. Set -sample- width to 2 (defines the number of bits required to represent the value on storage to 2)
5. read modified .wav file and transfer it to PCM form
6. Divide the audio into number of frames each of x millisecond(s) duration where x is a non-negative number

5.8.13 VOICE ACTIVITY DETECTION ALGORITHM (VAD)

Voice activity detection (VAD) is a technique in which the presence or absence of human speech is detected. The main idea behind VAD algorithm is detecting sudden changes in energy, spectral, cepstral distances, in order to satisfy different requirements from various features latency, sensitivity, accuracy, and cost.
For this project , Google's webRTC VAD will be used for voice detection.

Figure 11 WebRTC logo

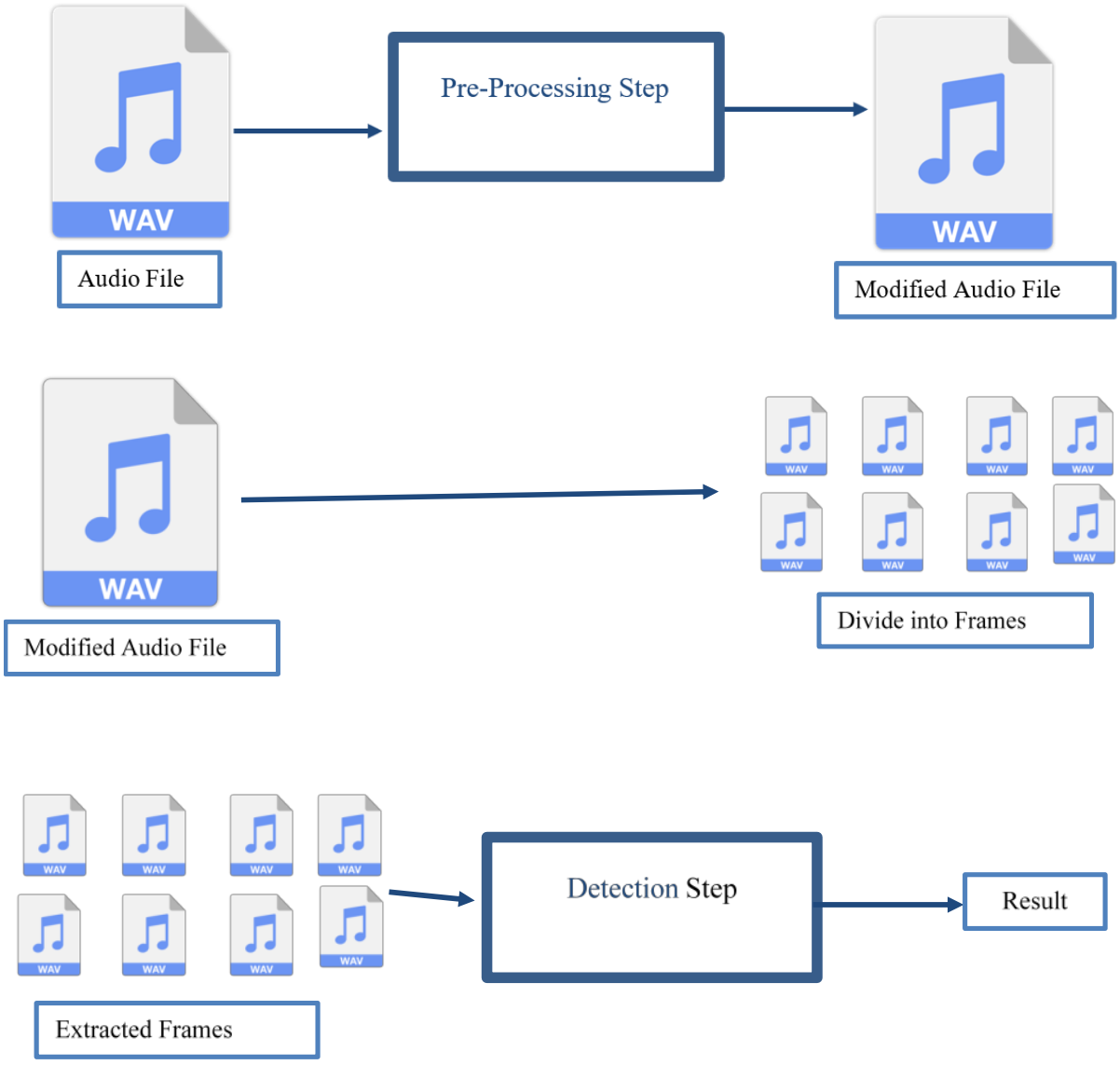
5.8.14 DETECTION PROCESS

1) Frames generated after pre-processing of audio is passed one by one to the VAD for detection , results of each of the frames is appended to one list (Python array like data structure)
2) The results list is then summed to know how many of the frames have voice in

a) For example if 3 frames have voice in them this means that 3*x milliseconds of audio file has voice in them

3) The last step is checking if Num of frames detected is greater than a given threshold if such this means this is a considered cheating case else no cheating case

5.8.15 PROCESS SUMMARY



5.8.10 MODEL INTERFERENCE ARCHITECTURE EVOLUTION

5.8.10.1 ARCHITECTURE VERSION 1

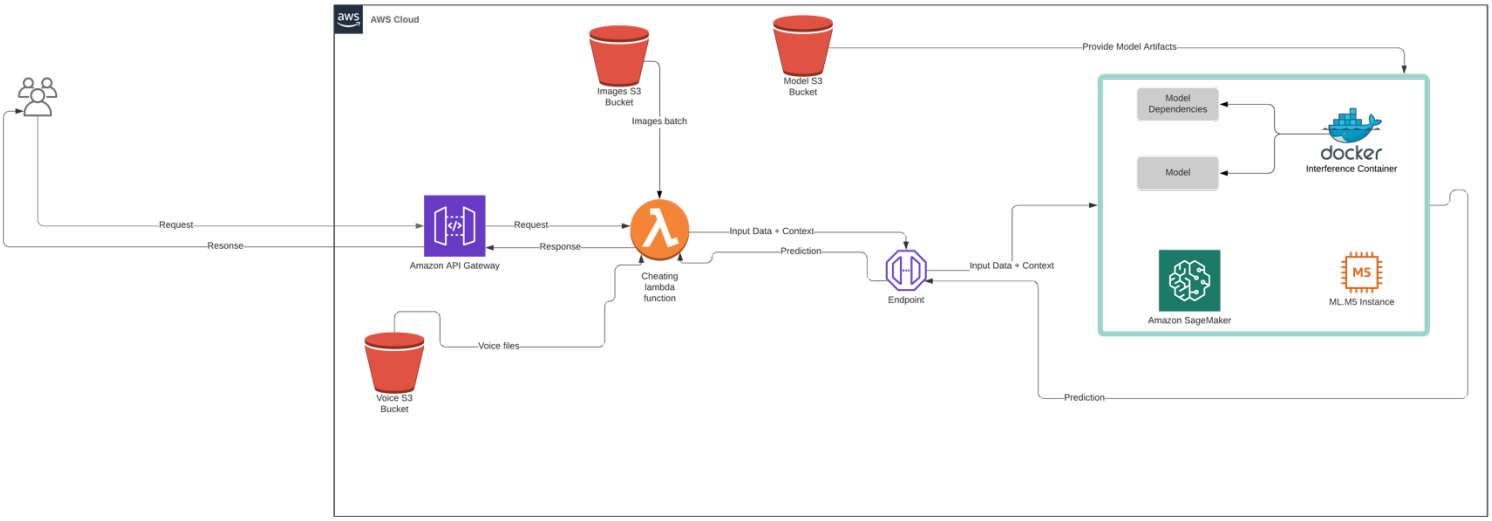


Figure 12 Cheating detection architecture 1

1. The Original Developed Architecture
2. All code has been added in one lambda function
   - YOLO object detection + Pose Estimation + Score Estimation + Voice activity Detection (VAD) + Final Scoring + return response
3. Performance has been moderate with Average running time of 45-55 Seconds

5.8.10.2 ARCHITECTURE VERSION 2

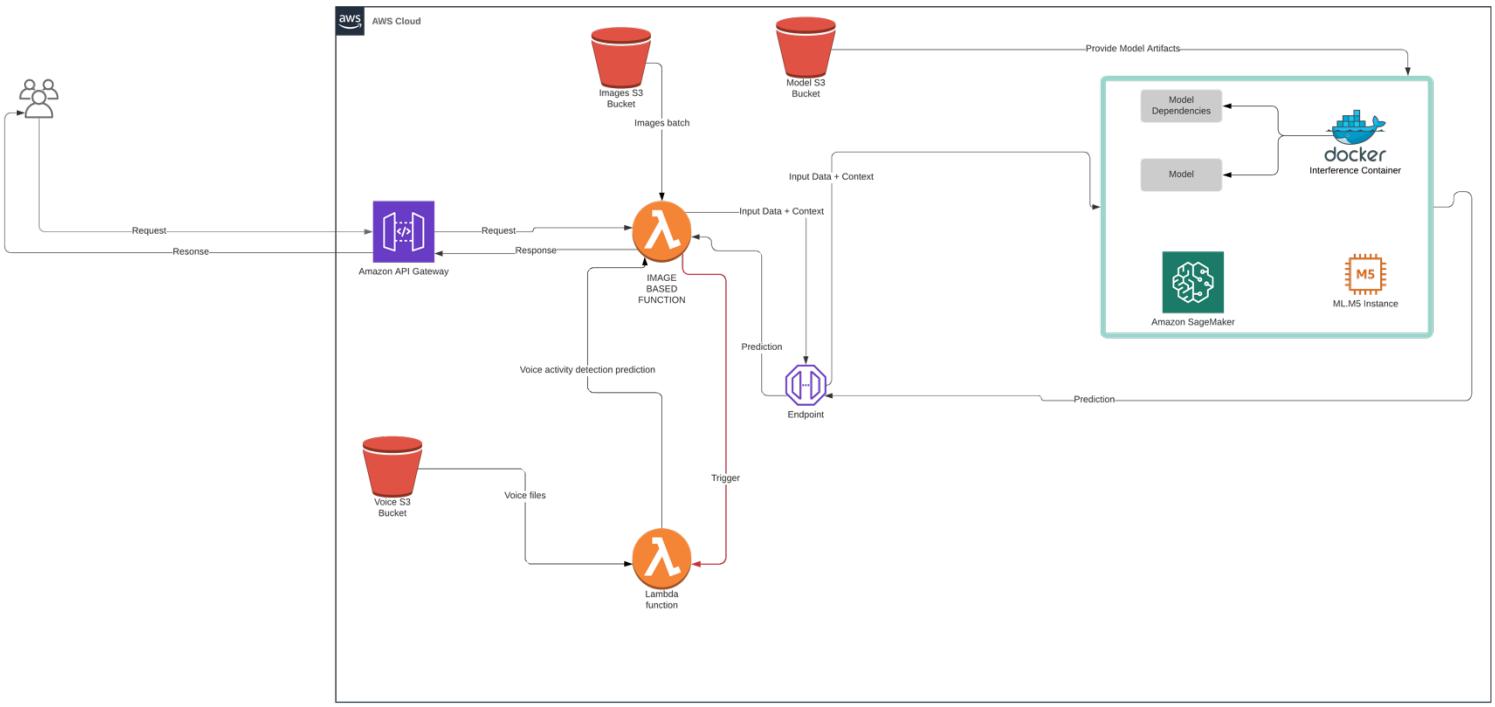


Figure 13 Cheating detection architecture version 2

1. Added new lambda function for processing audio
2. Triggered async by first lambda then results is fetched by first lambda
3. Voice activity Detection (VAD)
4. Image based function has been added in original lambda
5. YOLO object detection + Pose Estimation + Score Estimation + Final Scoring + return response
6. Performance has been moderate with Average running time of 35-40 Seconds

5.8.10.3 ARCHITECTURE VERSION 3

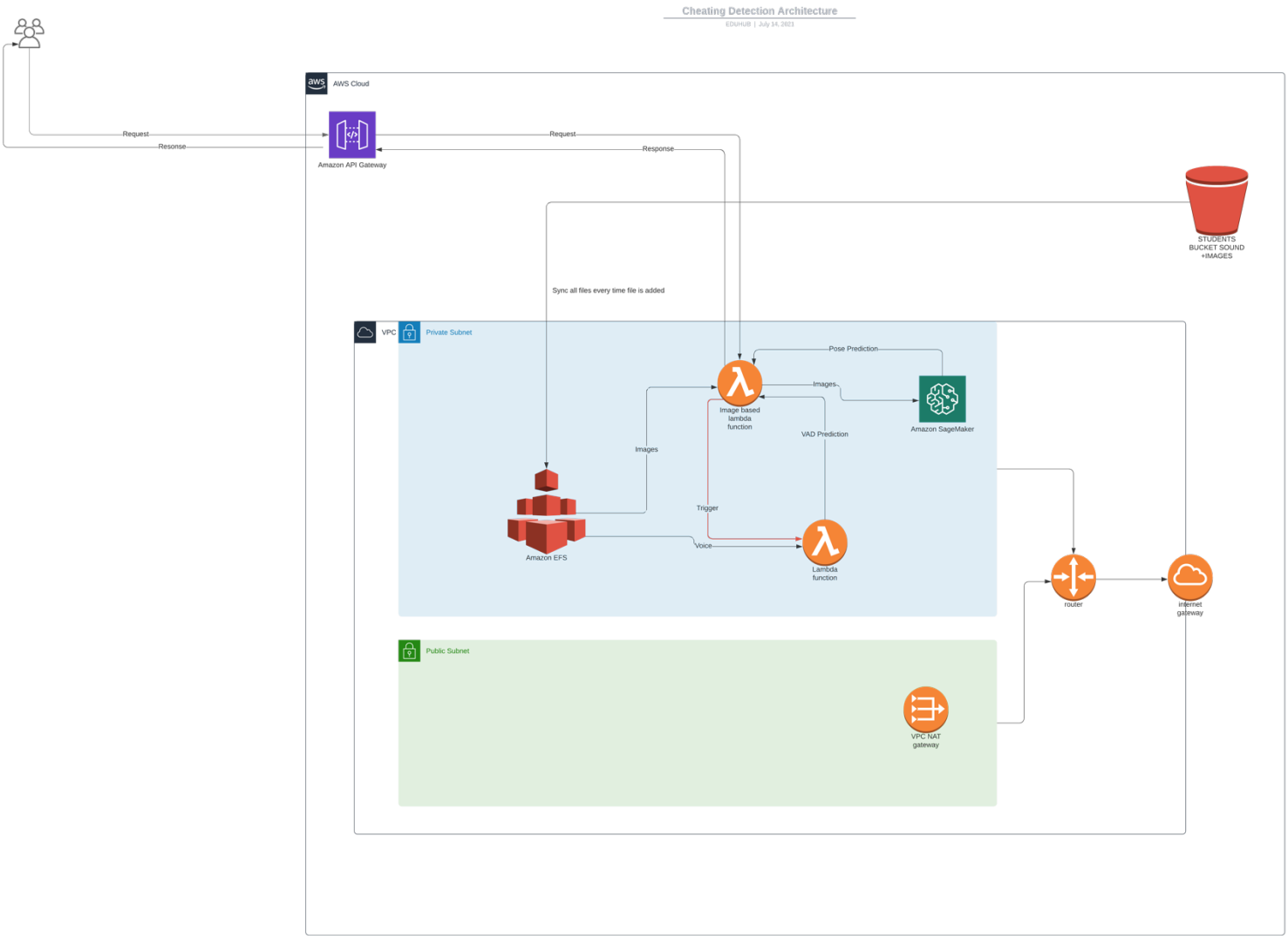


Figure 14 Cheating Detection architecture version 3

1. Architecture Redesigned to include :
    - Virtual Private Cloud with 2 subnets :
        - Private subnet : Containing all the lambda functions , SageMaker endpoint and Amazon Elastic File System

        - Public subnet : Containing Nat Gateway where all private subnet traffic is redirected to
    - Internet Gateway is added to enable communication with S3 (as VPC is internet isolated and cannot access it without internet gateway and nat gateway)
2. Newly added Elastic File System decrease latency of reading files from S3 and converting them to numpy array to read them , Images and voice files is synced to EFS where it can be read directly
- Every time an event happen to S3 a lambda function (not shown in above figure for simplicity and size constraints on word) is triggered that sync all S3 files with EFS
3. Performance has been moderate with Average running time of 28-40 Seconds

5.8.10.4 ARCHITECTURE VERSION 4

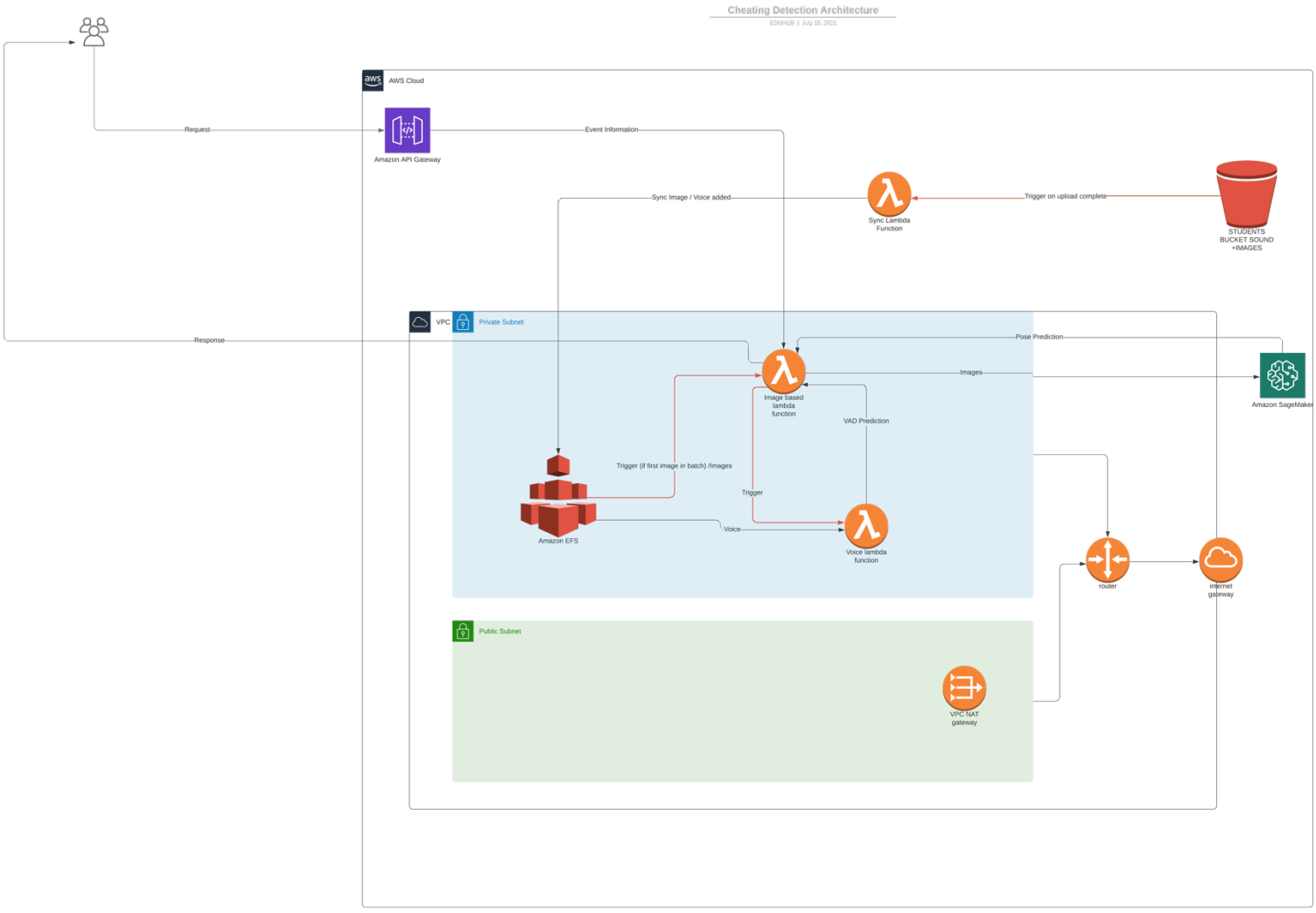


Figure 15 Cheating Detection architecture version 4

1. Added new lambda function to sync files once uploaded to S3 to EFS
    - Function also fires/trigger main lambda function that works on images if image uploaded is the first image of a batch

2. Performance has been moderate with Average running time of 23-35 Seconds

5.8.10.5 ARCHITECTURE VERSION 5

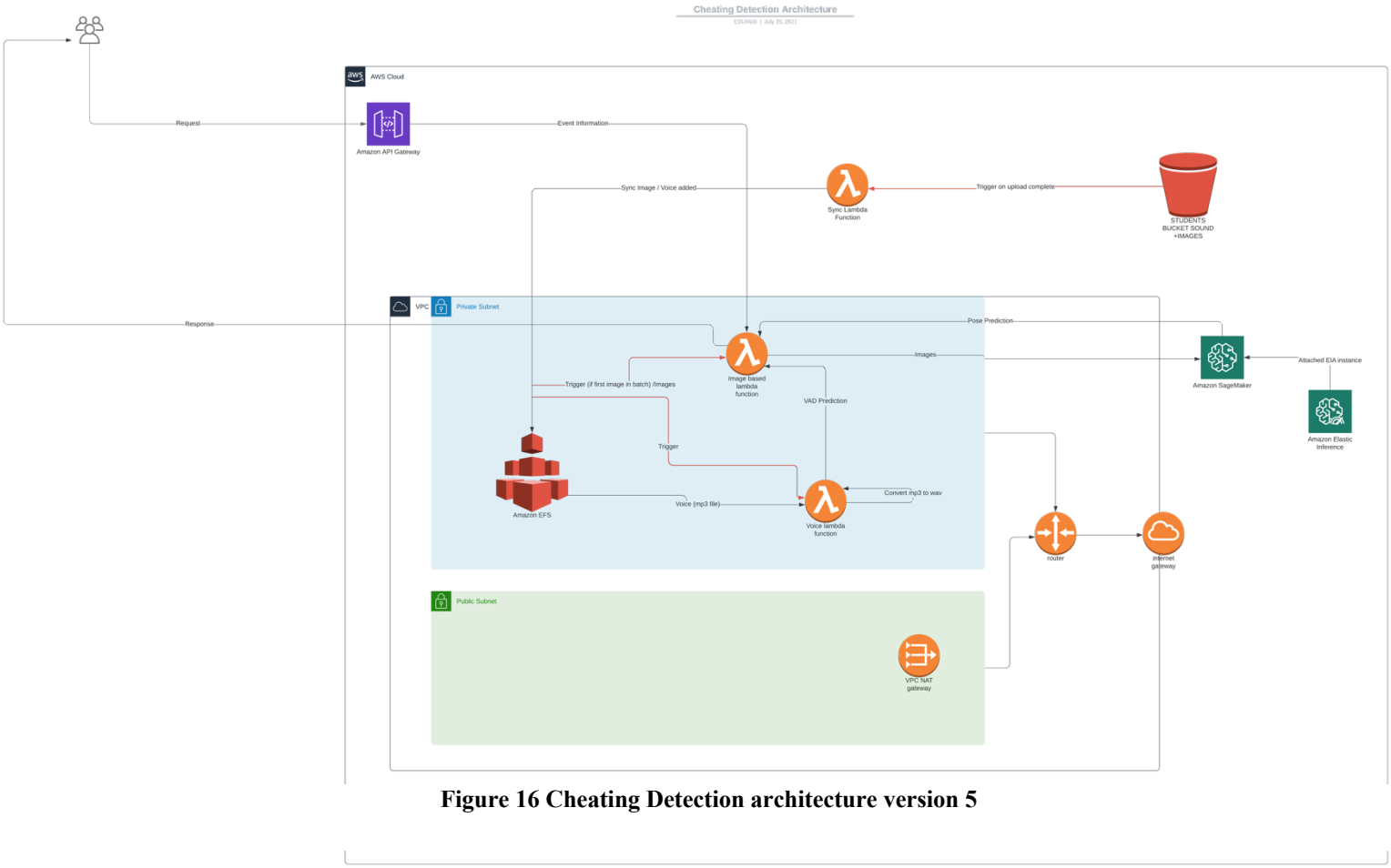


Figure 16 Cheating Detection architecture version 5

1. Added on fly conversion of lightweight mp3 files to wav files that can be worked on by the VAD Detection model
2. Sync Function now fires/trigger VAD lambda function that works on Voice if voice file (mp3) is uploaded
3. Added Elastic Interference (AWS EIA) to reduce the pose estimation run time
4. Performance has been moderate with Average running time of 14-20 Seconds

5.8.10.6 ARCHITECTURE VERSION 6

Same as Architecture version 5 but added handling of missed or dropped images by re-looping on missing images from original loop and retrying to read and process them if possible.

5.8.11 CHEATING REPORTING SYSTEM

5.8.11.1 INTRODUCTION TO REPORTING SYSTEM

Cheating Detection is a long and complex process that takes place on multiple stages , Cheating Detection uses Co-op functionality between App Front End , App Back End and also AWS server backend. The following figures Summarize some of the hidden complexities of implementation.

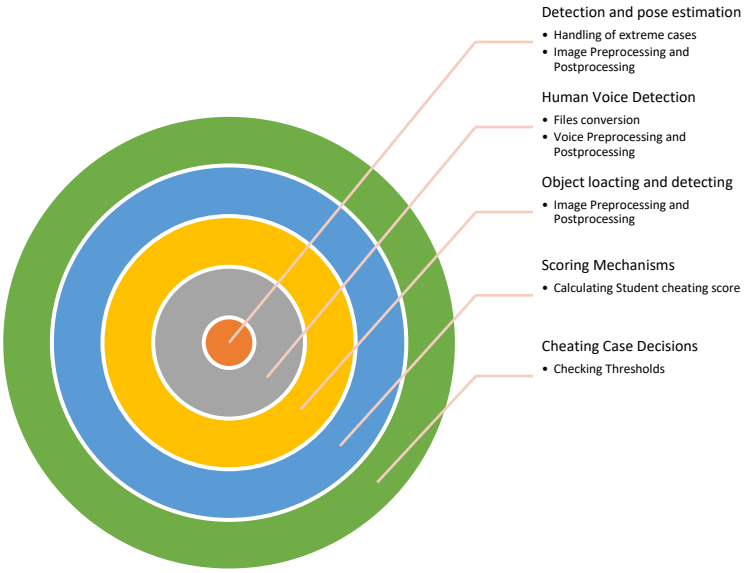


Figure 17 Complexities of Processing

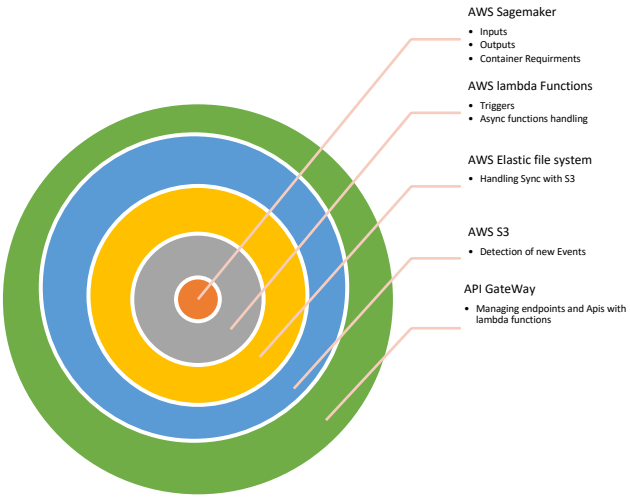


Figure 18 Server side (AWS) Complexities

From Above illustrations it was critical to create a user friendly explanation of each cheating case detected hidden all the complexities of the system and reporting the case in more less technical way. Since this System is targeting Educational based institutions it was decided to create 2 ways of notification one for students in shape of alerting pop-up notification and the other is a full report summarizing the cheating case and why the system consider it cheating.

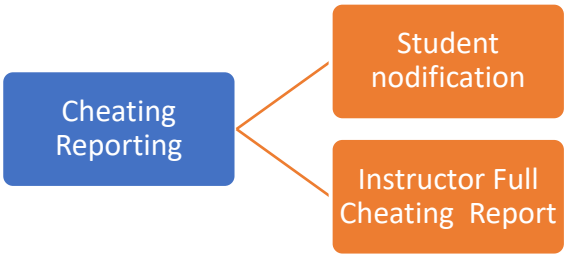


Figure 19 Cheating Detection Nidifications

5.8.11.2 REPORTING SYSTEM COMPONENTS

The Report created is a pdf file consisting of 5 Pages which are :

- Page 1 (Intro)
  - Student Name
  - Reporting time
- Page 2 (Overview)
  - Cheating case overview
  - Displaying scores and which cheating factor is considered for this cheating case

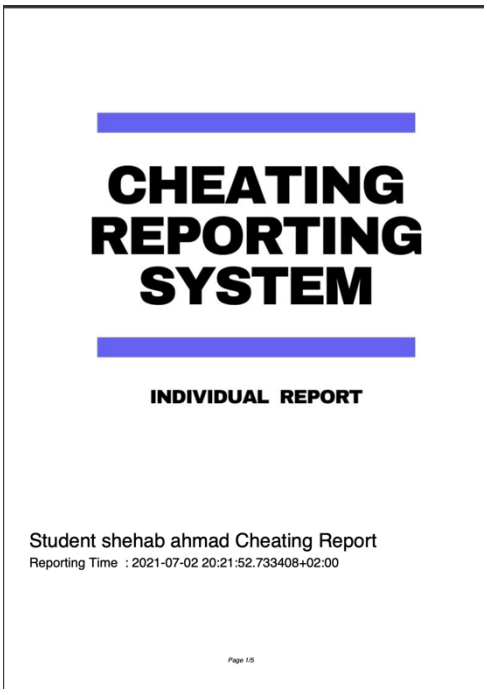


Figure 20 Cheating Report Page 1

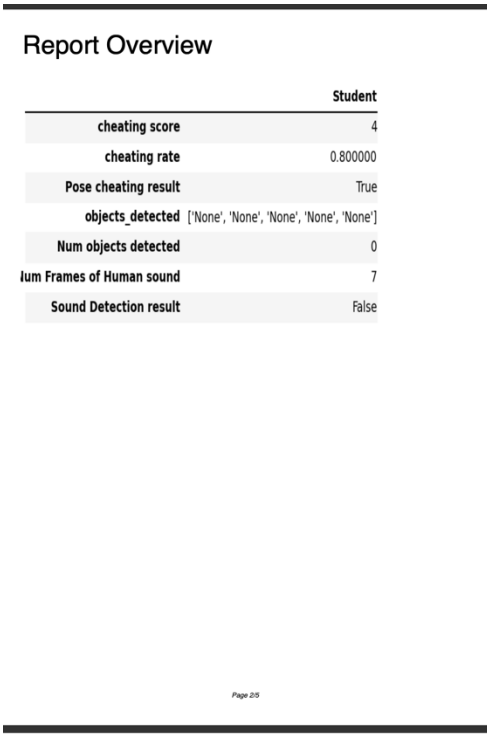


Figure 21 Cheating Report Page 2

The Report created is a pdf file consisting of 5 Pages which are :

- Page 3 (Images)
  - 4 Images of the batch
  - Most suspicious ones
- Page 4 (Graphical Overview of Pose Estimation Scores)
  - Horizontal and Vertical Pose Line and bar graphs
  - 2 Circle pies summarizing cheating in vertical and horizontal pose
  - A combined line graph of two poses
  - Heatmap of Poses scoring
- Page 5 (Voice Activity Detection)
  - Including a waffle chart for VAD predictions

Figure 22 Cheating Report Page 3

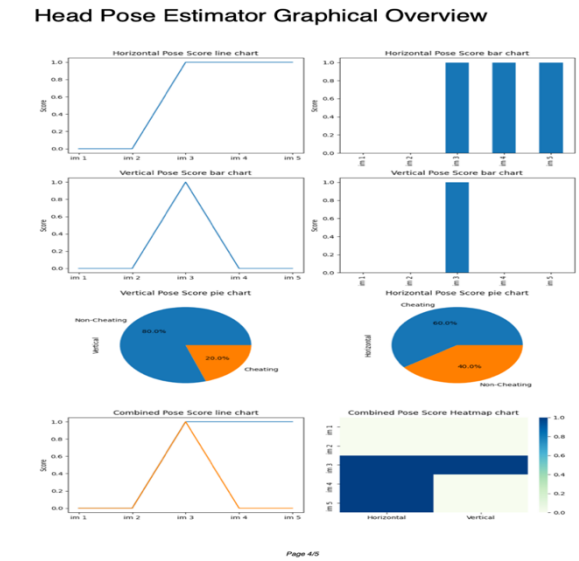


Figure 24 Cheating Report Page 4

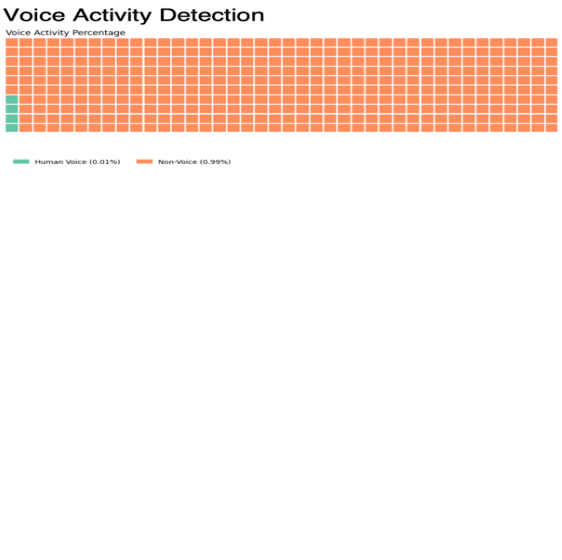


Figure 24 Cheating Report Page 5

5.8.11.3 REPORTING SYSTEM IMPLEMENTATION

- Matplotlib is used to create subplots (4 or 8) each plot of them is representing either an image or graph (line / bar / pie)
- Pandas was used in graphs to provide data to plot (Data gathered from cheating detection functions)
- Seaborn was used to create the heatmap
- Overview DataFrame is created exported as image and placed in overview page
- WafflePlot package is used to create a waffle chart in VAD page (last page)
- FPDF package was used to wrap all previous in one pdf file
- That pdf is placed on EFS
- Then SES lambda function is triggered with the path of the pdf file
- SES create SMTP friendly format merging the Body of email with the attachment (pdf report)
- SES lambda fires request on SES
- SES send Email to the instructor

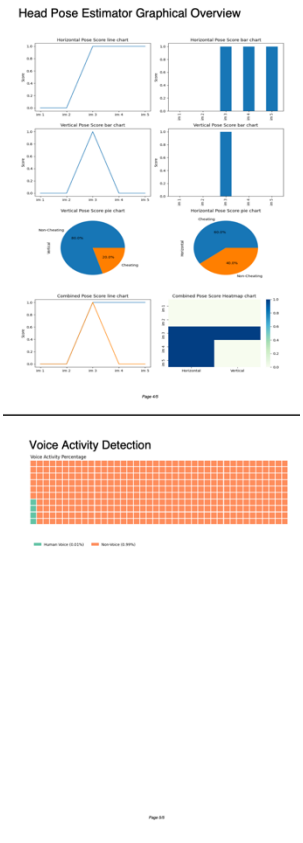


Figure 25 Cheating Detection Report Pages 2-5

5.8.11.5 REPORTING SYSTEM DELIEVERY METHOD INTRODUCTION

- Pdf report of cheating case is delivered to the instructor by Email , whenever a cheating case occur the provided email is delivered the pdf along with the name of student accused of cheating.
- Amazon SES service is used to register instructors and send them Emails whenever cheating case is predicted

Figure 26 Amazon SES service

5.8.11.5 REPORTING SYSTEM DELIEVERY METHOD PROBLEMS

- Amazon SES uses Simple Mail Transfer Protocol (SMTP) which uses a standard which state that every email message consists of a header and a body. The header consists of message metadata, and the body contains the message itself.
- The SMTP protocol was originally designed to send email messages that only contained 7-bit ASCII characters. This specification makes SMTP insufficient for non-ASCII text encodings (such as Unicode), binary content, or attachments.
- Solution : Using Multipurpose Internet Mail Extensions standard (MIME) which was developed to make it possible to send many other kinds of content using SMTP.The MIME standard works by breaking the message body into multiple parts and then specifying what is to be done with each part. For example, one part of an email message body might be plain text, while another might be HTML. In addition, MIME allows email messages to contain one or more attachments. Message recipients can view the attachments from within their email clients, or they can save the attachments. The message header and content are separated by a blank line. Each part of the email is separated by a boundary, a string of characters that denotes the beginning and ending of each part.

5.8.11.5 REPORTING SYSTEM DELIEVERY METHOD EXAMPLE

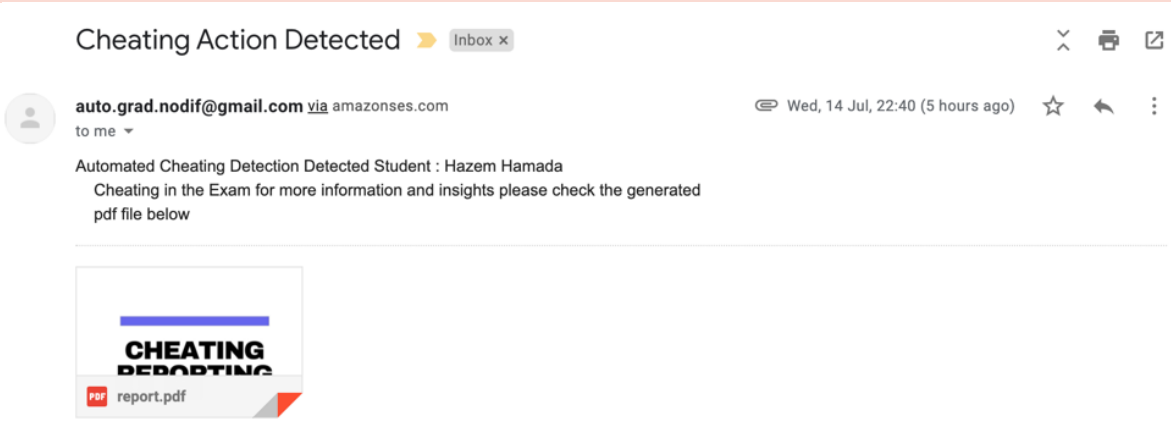


Figure 27 Cheating Detection Email

5.8.11.5 REPORTING SYSTE DEPLOYMENT ARCHITECTURE

- Lambda function (create report function)is created to create the pdf assets plots/graphs and export them to EFS , after which assets is used to create the pdf
- Pdf is exported to EFS
- Second Lambda function (create and send email function)is then triggered with the path

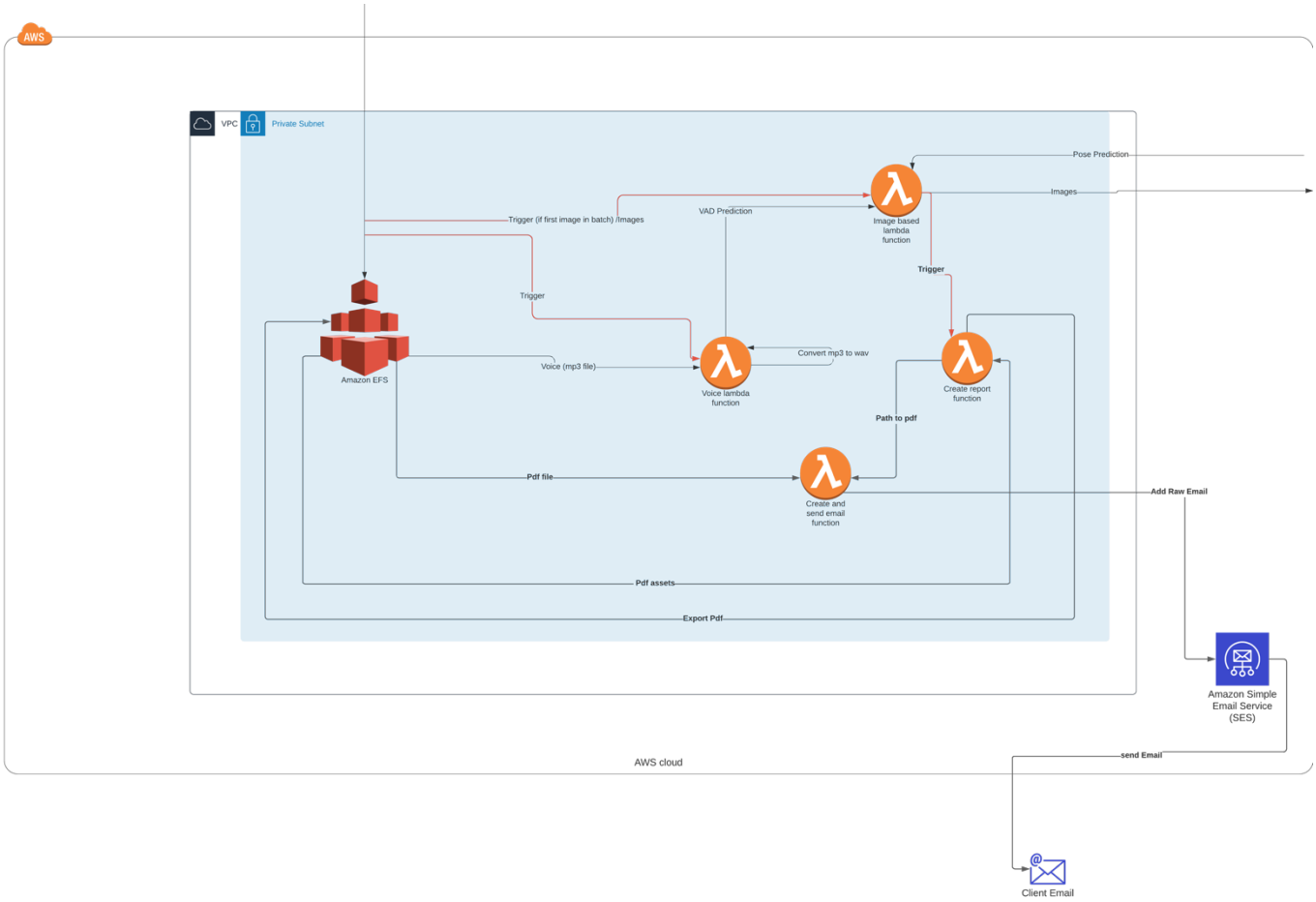


Figure 28 Reporting Architecture

to pdf , using MIME package an email is created with attached pdf file

- Second Lambda function triggers SES with the Email
- SES Send the email to instructor/Client

5.8.12 MEETING NON-FUNCTIONAL REQUIRMENTS

In this section of model documentation model is tested against non-functional requirements as :

- Scalability (as in text moderation model)
- Reliability and Availability (as in text moderation model)
- Performance
- Security

5.8.12.1 PERFORMANCE

Performance has been the main target for us , Providing a near Realtime environment for cheating detection was critical for the process. By using Architecture that consist of scalable high performance asynchronous entities that work together to ensure highest possible speed

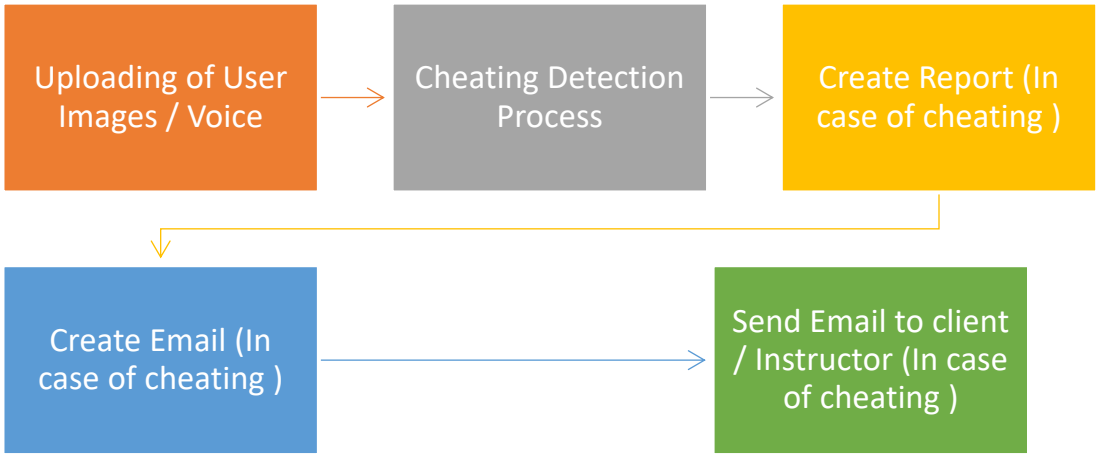


Figure 29 Cheating Detection Processing Overview

processing time.

However Performance in Cheating detection model is affected by user's internet speed , Best case scenario is achieved if uploading speed is in range 50-60 Kbps.

Performance can was incremented in Cheating Detection Process by :

- Adding Elastic interference to enhance the speed of pose estimation
- Increase the memory to lambda function to 8-10 GB to increase the speed of execution
- Adjusting EFS to Brust Mode]

Performance was increased in Reporting system by :

- Increasing lambda memory
- Attach to adding 4 Images plots instead of more

Performance was increased in Creating Email by :

- Increasing lambda memory

- Using lightweight Multipurpose Internet Mail Extensions standard

Performance was increased in email sending by :

- Using AWS's SES service that uses queues in case of high traffic to maintain performance

5.8.12.2 SECURITY

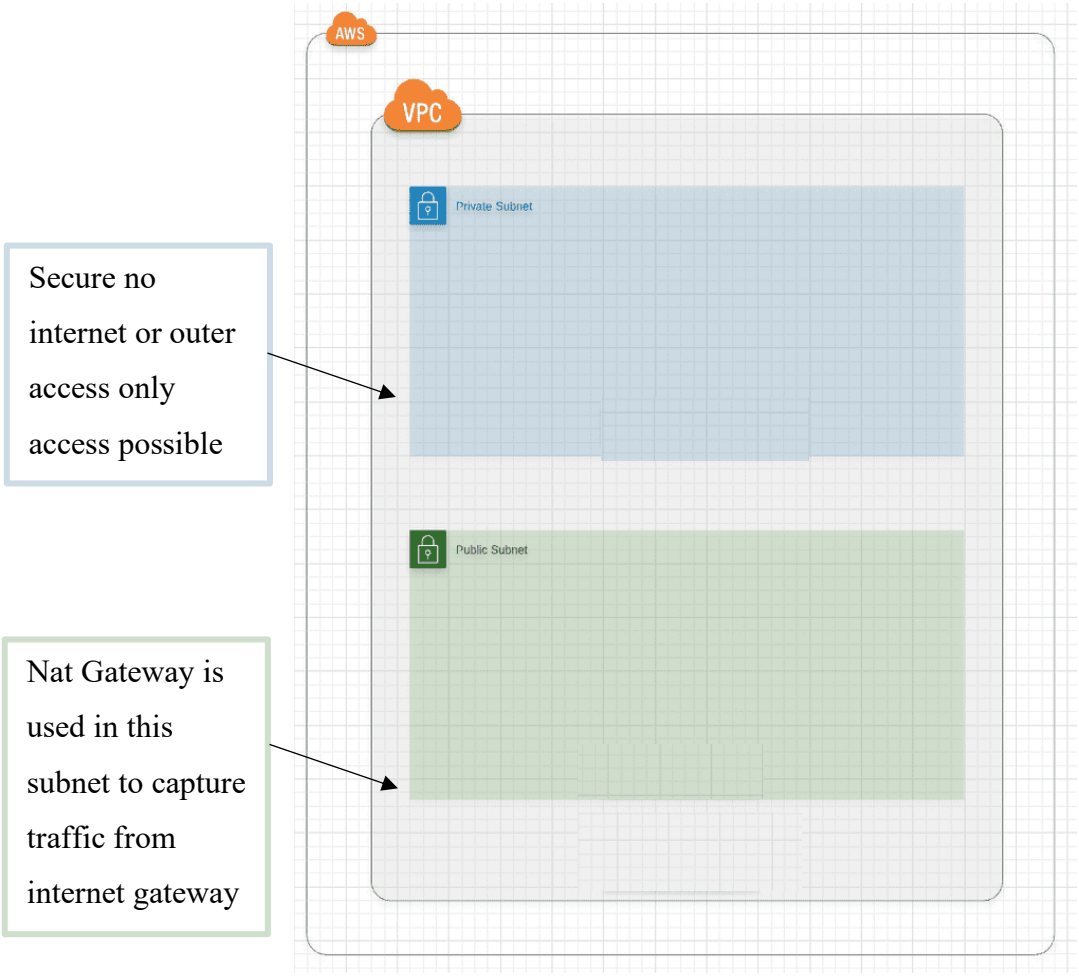


Figure 30 Virtual Private Cloud Architecture

- Security in cheating detection is greatly enhanced by using Virtual Private Cloud (VPC)
- Amazon Virtual Private Cloud (Amazon VPC) is a service that launch AWS resources in a logically isolated virtual network.
- Giving complete control over virtual networking environment, including selection of IP address range, creation of subnets, and configuration of route tables and network gateways.

- Amazon VPC can create a public-facing subnet for your web servers that have access to the internet.
    - Service included in cheating detection VPC public subnet is NAT Gateway
- It also lets you place your backend systems, such as databases or application servers, in a private-facing subnet with no internet access.
    - Service included in cheating detection VPC private subnet is EFS and lambda functions
- This means that there's no access to any student data within the system (EFS) from any kind of out-of-AWS services.
    - This ensure the confidentiality of the data and its security
- Traffic needed is sent from different AWS to public subnet NAT gatway from internet gateway and is directed to private subnet for resources there to use it

5.8.13 TEST CASE

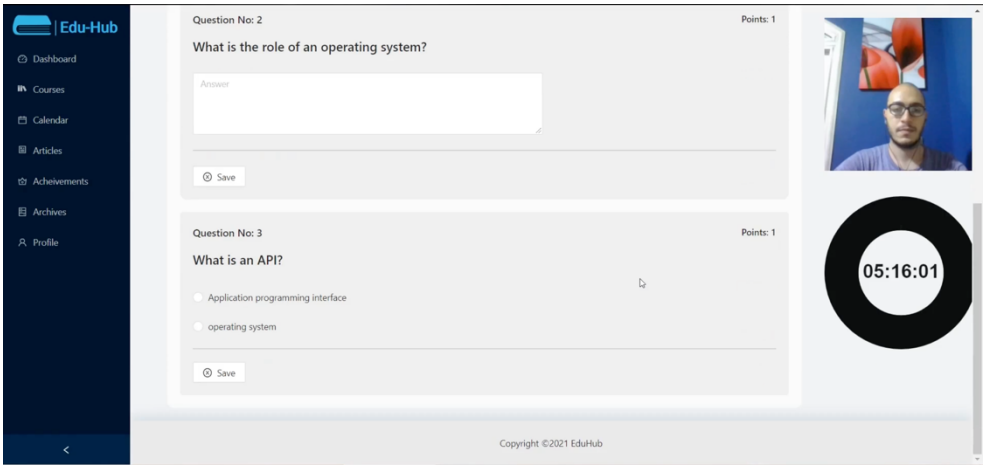


Figure 31 Student Taking Exam (No Cheating)

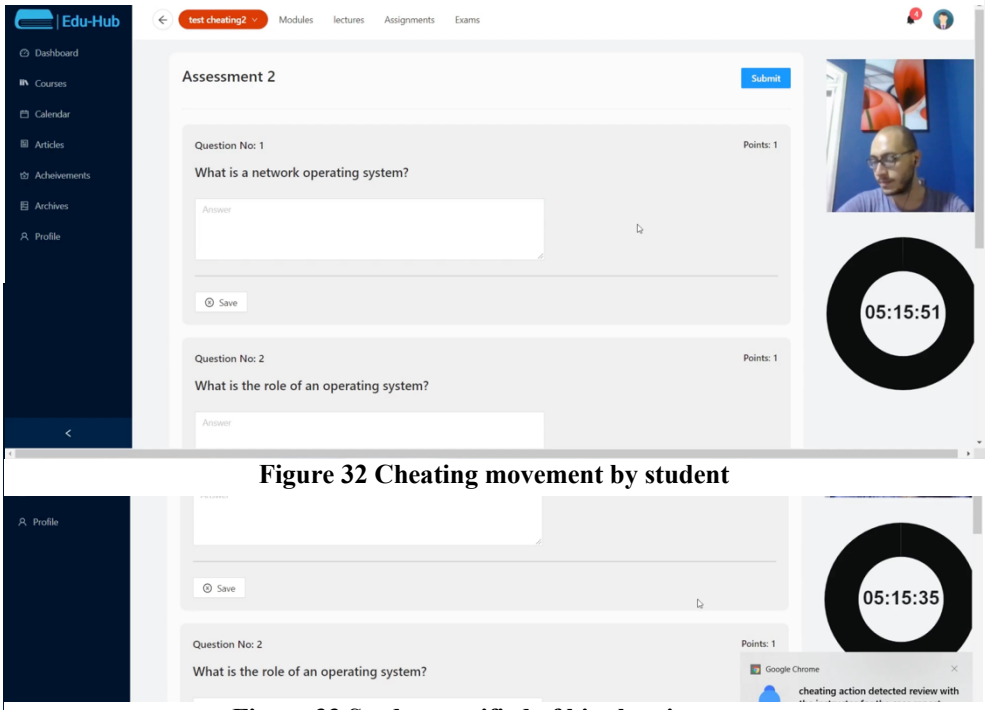


Figure 32 Cheating movement by student

Figure 33 Student notified of his cheating case

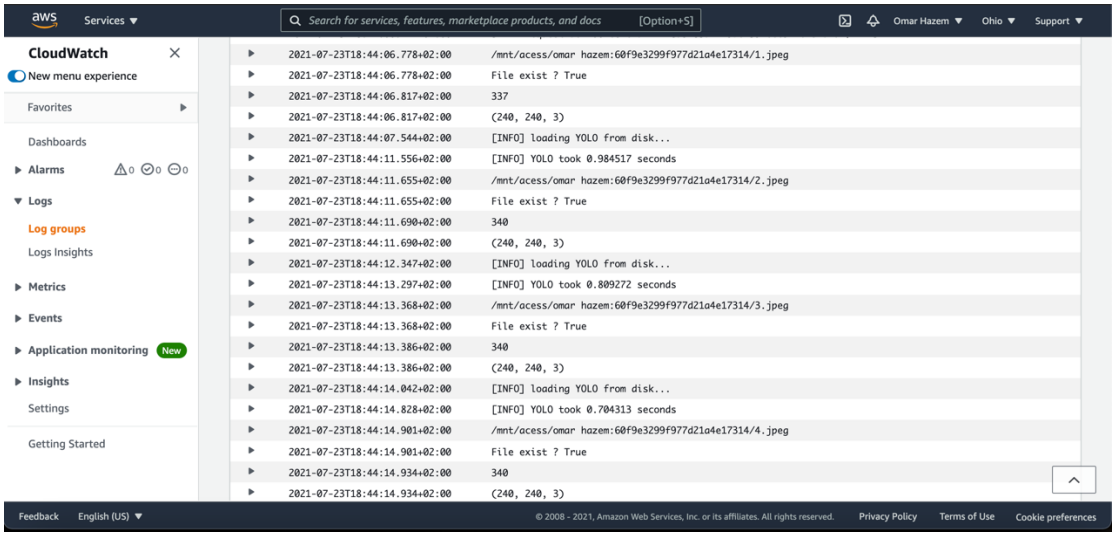


Figure 34 AWS Logs of Processing

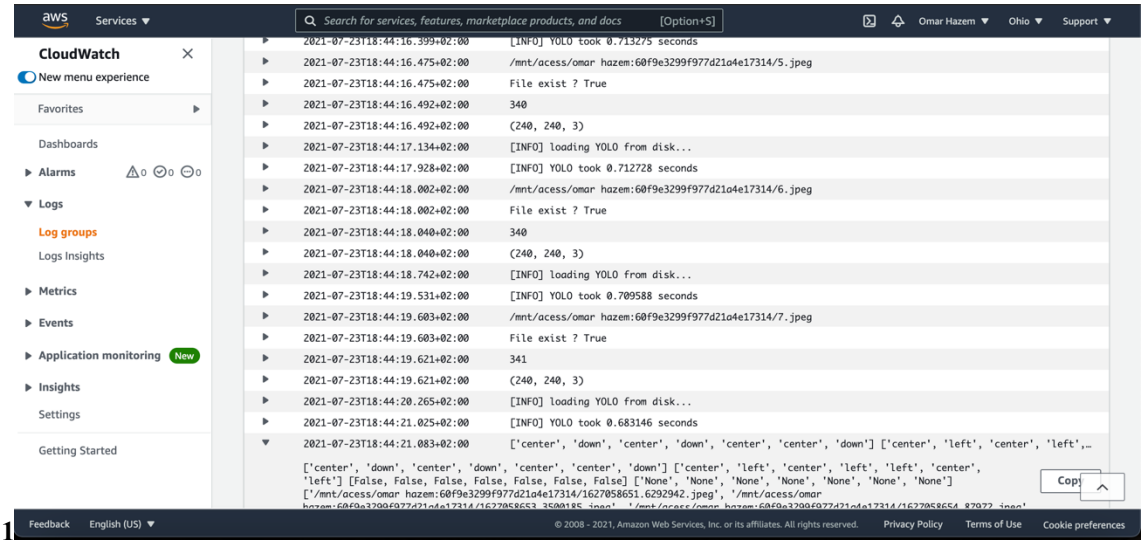


Figure 35 AWS Logs of Processing 2

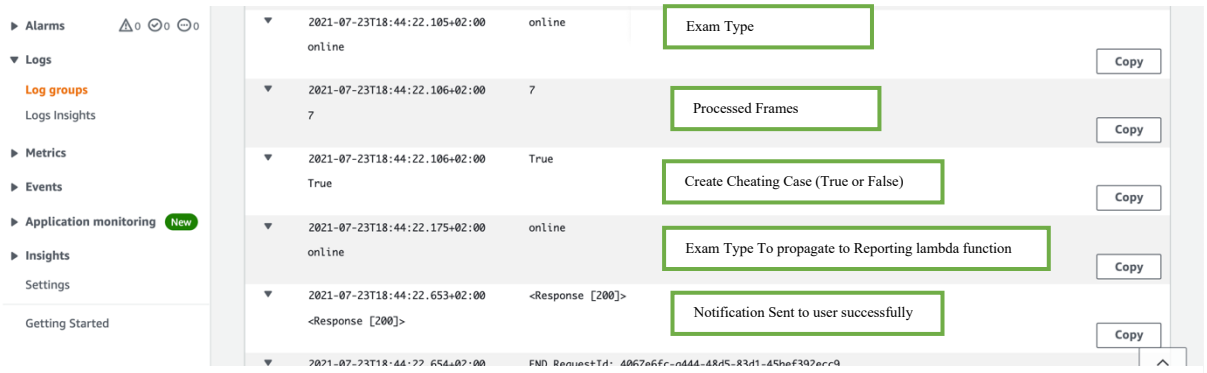


Figure 36 AWS Logs of Processing 3

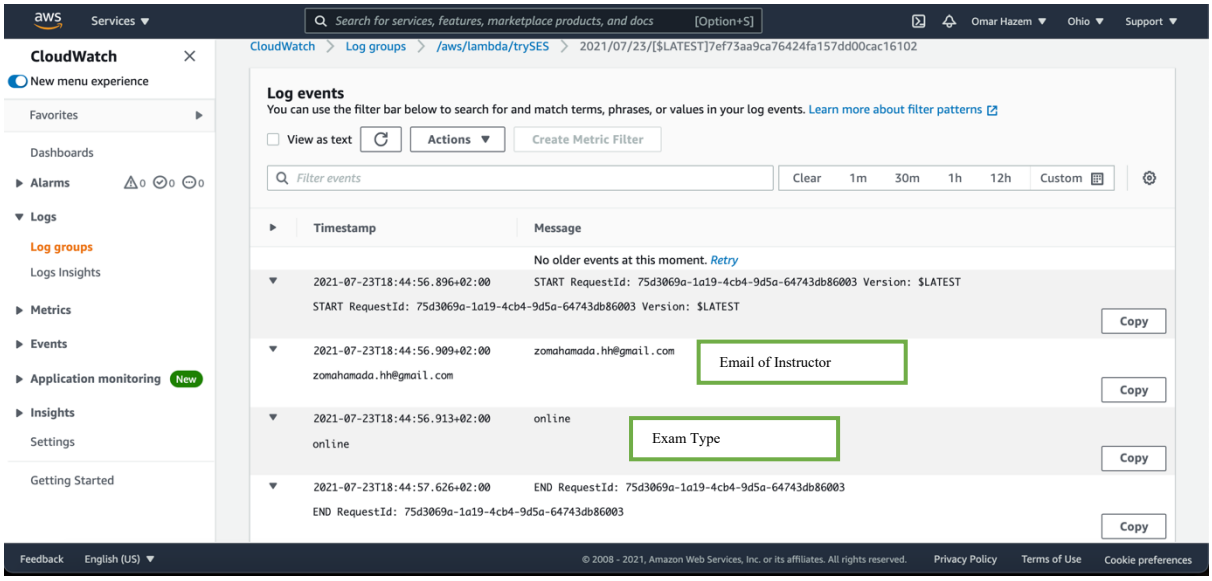


Figure 37 AWS Logs of Email Creation Processing

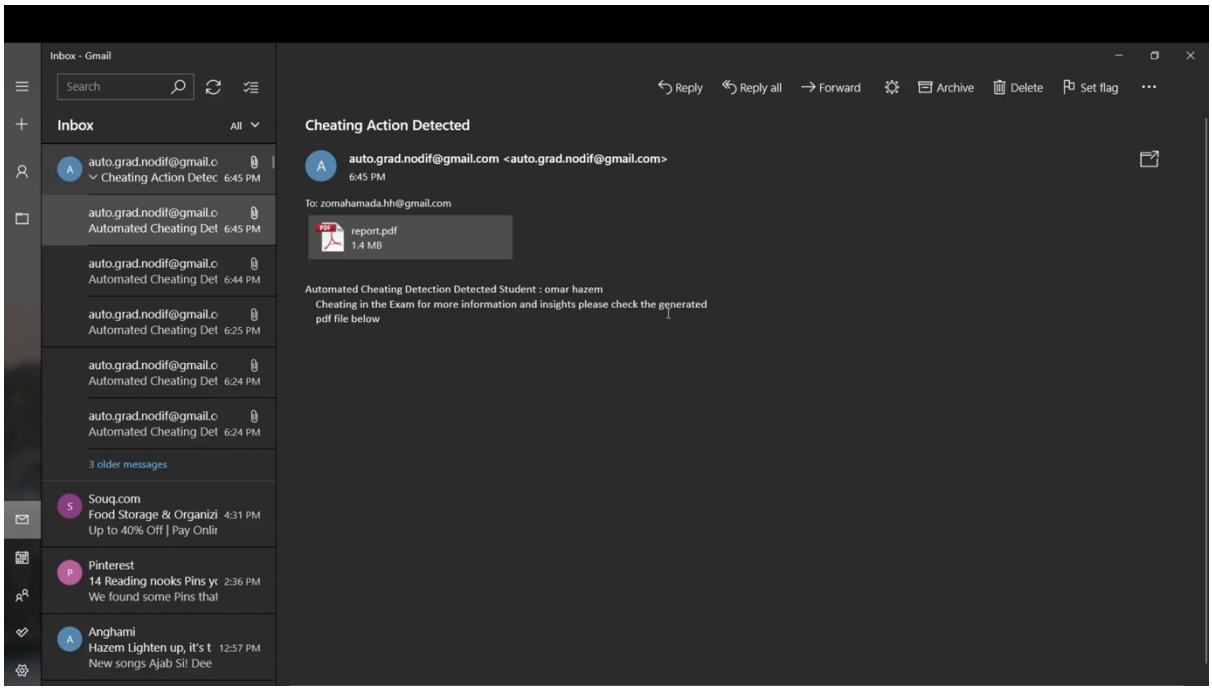


Figure 38 Client Email with email delievered