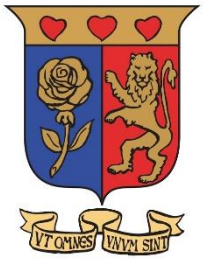

Strathmore
UNIVERSITY

Online Exams Invigilation Aid through Analysis of Direction of Visual Attention with Convolutional Neural Networks

Waithaka, John Gachihi

101892

Supervisor: Derdus Mosoti

An informatics system project documentation submitted to the Faculty of Information Technology in partial fulfilment of the requirements of the award of a Degree in Informatics and Computer Science

Date of submission: February 2021

Declaration

I declare that this project has not been submitted to Strathmore University or any other University for the award of a degree in Informatics and Computer Science or any other degree.

Admission Number: 101892

Signature: _____ Date: _____

I certify that this work is being submitted for examination with my approval.

Supervisor name: Derdus Mosoti

Signature: _____ Date: _____

Abstract

The rate of adoption of online learning has been rising over the years. This rate has recently spiked due to the outbreak of the COVID-19 pandemic. This has resulted in a need for conducting exams online. However, online exams have the disadvantage of being costly to invigilate because of the high labour requirement of the current way of online invigilation, which involves human invigilators monitoring candidates using video streamed from the candidates' webcams. This work develops an automated online invigilation aid that uses Convolutional Neural Networks to analyze candidates' direction of visual attention (the direction one's sight is focused) to detect and report when a candidate might be cheating. The developed system is expected to reduce the effort and labour required to invigilate online exams by assisting human invigilators to detect cheating and, therefore, reduce the cost of online invigilation and online exams.

Table of Contents

Table of Figures

1 Introduction

1.1 Background

Over the years, the adoption of online learning has been rising around the world due to its conveniences (Allen and Seaman, 2007). For example, in the United States, the number of students in higher education taking at least one online course grew from 9.6% in 2002 to 29.7% in 2015 (Woldeab et al., 2017). Recently, the COVID-19 pandemic made online learning a necessity due to the government-mandated closure of schools in most countries in the world.

Further, even before COVID-19, there have been certain people who have not had easy or any access to the traditional form of learning. Some of these people include those living in poverty, those living in remote areas, and those living with disabilities. Online learning is regarded by some as a suitable means for providing better access to education to these people (Khan and Williams, 2006).

Assessment is an integral part of learning (McDowell and Sambell, 2014) – whether online or traditional. Learning depends on the feedback obtained from assessments (Berry, 2008). Assessment is, however, faced with the major problem of cheating. In a survey conducted among university students, 73% of 1,330 participants self-reported to have cheated in the past (Freiburger et al., 2017). Another study, though with a small sample of 30 students, found that 67% of the students had cheated in a test or exam (Balbuena and Lamela, 2015). Even a survey conducted in 1998 had 64% of 1,793 students admitting to having cheated (Mccabe et al., 2001).

To curb this problem, learning institutions use exam invigilation. This involves having a person monitor candidates taking an exam to make sure there is no cheating. This technique has proven its effectiveness in both traditional and online exams (Kerkvliet and Sigmund, 1999; Prince et al., 2009).

Invigilation in online exams (or online invigilation) is still a new and relatively untrusted area. However, the growth of and increased necessity for online learning is causing an increased need for online examinations and, consequently, an increased need for effective and efficient online invigilation.

Currently, the most prevalent form of online invigilation is human invigilation. This is where a human invigilator monitors candidates from a live video stream or stored recording taken

from the candidates' webcams. There are many commercial services available that offer this service. Some prominent ones are ProctorU and Kryterion.

The shortcoming of this form of invigilation is that it is significantly more costly than human invigilation in a traditional exam setting (Chuang et al., 2015, 2017). It requires more manpower, time, and money. This is probably owing to the higher optimal invigilator-to-candidate ratio in online invigilation, i.e., effective online invigilation requires a higher number of invigilators than physical invigilation for the same number of candidates. One invigilation software provider, for example, uses a 1 to 4-10 invigilator-to-candidate ratio (Software Secure, 2016) which is significantly higher than it is in physical invigilation.

This work contributes an automated invigilation aid that uses convolutional neural networks (CNN), to detect and report potential cases of cheating by analysing the direction of candidates' visual attention from images taken from their webcams.

It is expected that the automated invigilation aid will reduce the effort required from human invigilators in invigilating online exams, thus lowering the optimal invigilator-to-candidate ratio. This should reduce labour cost and, in some cases, time consumption, while also increasing the scalability of online invigilation.

1.2 Problem Statement

The current method of online invigilation – human invigilation – requires a lot of manpower to be effective. Going by Software Securer's invigilator-to-candidate ratio of about 1 to 10, an exam session of 500 candidates will require 50 invigilators. This is significantly higher than what is expected in physical invigilation for an exam room of 500 candidates.

Because of its high labour requirement, it is financially costly. Also, where recordings of candidates are stored and examined later, it becomes very time-consuming.

1.3 Aim

To develop an online exams invigilators' aid that automatically detects and reports, in real-time, possible cases of cheating by analysing each candidate's direction of visual attention from images taken from their webcam as they do an exam.

1.4 Objectives

I. To investigate the current state and challenges faced in online invigilation.

II. To identify indicators of cheating that can be captured by a webcam.
III. To identify and review CNN architectures that can learn to detect the identified indicators of cheating.
IV. To train and test a CNN cheating detection model based on the most suitable architecture identified in the preceding objective.
V. To design a basic online invigilation application that uses the trained cheating detection model to detect and report possible cases of cheating.
VI. To develop and test the online invigilation application.

1.5 Research Questions

I. What is the current state of, and the challenges faced in online invigilation?
II. Which indicators of cheating can be captured by a webcam?
III. Which CNN architectures can best learn to detect these indicators of cheating?
IV. How can one train and test a CNN model to detect the indicators of cheating?
V. How can one design a basic online invigilation application that uses the trained CNN model to detect and report possible cases of cheating?
VI. How can one develop and test the online invigilation application?

1.6 Scope of the Study

First, this work focused on detecting cheating by analysing a candidate's direction of visual attention from images taken from a candidate's webcam. Other inputs, such as audio and typing patterns, were not considered.

Second, the kind of exams referred to in this work are strictly those which are fully presented and taken on a laptop or desktop. Additionally, it refers to exams where reference material such as notebooks, textbooks and mobile phones are prohibited.

Third, the kind of cheating that this work focused on was the kind that involves the use of cheating material that is external to the device used to take an exam, for example, *cheat-sheets* and notebooks. Cheating done using the device used to take the exam, for example, browsing, was not considered as systems already exist that effectively prevent this kind of cheating.

Lastly, the study focused on detecting limited cheating tactics, specifically, where a candidate reads something to the left or right of their exam device, or beneath their desk.

1.7 Justification

A prime cause of the reluctance of learning institutions to adopt online learning is the lower trust and credibility in online exams than in traditional exams. Solutions for effective, efficient, and scalable online invigilation may significantly improve the confidence in, and credibility of online exams. That would result in greater adoption of online learning among learning institutions which will have the following beneficial effects.

First, greater adoption of online learning by learning institutions will create a more robust education system. It will be more robust such that the closure of schools for reasons such as the recent COVID-19 pandemic will not hinder learning. The lack of physical access to school facilities will be a non-issue as online learning will provide an adequate fallback.

Secondly, the groups of people like the poor, the disabled and those living in remote areas will have better and more convenient access to education, therefore lowering the educational inequality in society.

2 Literature Review

2.1 Introduction

This chapter reviews the literature on the following topics. First, section 2.2 reviews the literature on the current state of online invigilation and the challenges it is facing. Section 2.3 then reviews the literature on indicators of cheating that are detectable using webcam images. Section 2.4 reviews literature on relevant CNN architectures. Section 2.5 and section 2.6 analyze work related to automated online invigilation and the gap in these works.

Lastly, a conceptual framework of the proposed system is provided in section 2.7.

2.2 The Current State of Online Invigilation and Challenges faced

Human invigilation is the most prevalent form of online invigilation as mentioned by Asep and Bandung (2019), and Atoum et al. (2017). It involves a human invigilator monitoring candidates using video taken from the candidates' webcams. The invigilation is done live or by reviewing saved video recordings.

There are many commercial online human invigilation systems. Some of these are ProctorU, Kryterion, Examity and Ulearn.

A major challenge faced in the invigilation of online exams is that it is constrained by the field of view of the camera used to observe candidates. Typical laptop webcams have a small field of view. They can, usually, only capture the head and shoulders of a user. Therefore, invigilators have a limited view of a candidate's environment (Chuang et al., 2017). They cannot, for instance, see whether there is a 'cheat sheet' on a candidate's desk or a phone beneath their desk. Therefore, to detect that a candidate is cheating, invigilators solely rely on the candidates' head movement, eye movement and other such features that can be captured by a webcam.

Because of this limitation, invigilators are required to pay more attention to each candidate than is needed in physical invigilation. Therefore, as the number of candidates an invigilator is required to invigilate increases, his effectiveness reduces at a higher rate than in physical invigilation. For this reason, the optimal candidate-to-invigilator ratio for online exams is usually much lower than in traditional exams. Consequently, invigilation of online exams requires more labour, cost and, possibly, time.

2.3 Indicators of Cheating

This section contains a review of the literature on features that have been found to indicate or identify cheating and how they can be measured.

2.3.1 Early Studies on the Identification of Cheating

Most early work on the indicators of cheating considered a variety of factors which Crown and Spiller (1998) grouped into two categories – individual and situational factors. Individual factors were candidates' personality features such as age, gender, religion, grade and personality. Whereas situational factors were contextual features of candidates' academic institutions such as the academic institution's honour code, value counselling, surveillance, and sanctions or threat of punishment.

However, as stated by Chuang et al. (2017), these factors are best used to determine the likelihood of cheating in a population. Though they may be used for estimating the probability of the occurrence of cheating, they cannot, by themselves be used to detect cheating as it is taking place. For this reason, these factors were not considered in this work.

2.3.2 Visual Focus of Attention/Direction of Visual Attention

Chuang et al. (2017) describe the visual focus of attention (VFOA) as, "the particular location in one's visual field where a person focuses in the attentive mode." It is determined by a person's eye gaze, i.e., the direction the eyes point to, but can also be estimated using head pose (Ba and Odobez, 2009; Smith et al., 2008).

Chuang et al. (2017) were able to statistically show that a candidate's VFOA can significantly detect possible cheating incidents. Their rationale for using VFOA as an indicator of cheating is as follows. During an online exam, candidates' visual attention, in normal circumstances, is focused on their screen. For them to cheat, for example, by reading from a *cheat-sheet*, their visual attention will deviate from their screen. Therefore, when such a deviation is detected, it is possible (but not guaranteed) that the candidate is cheating.

This idea is corroborated by an experienced online exam invigilator who, in a news article, said, "When the eyes start veering off to the side, that's clearly a red flag." (Steve Kolowich, 2013).

This is the main indicator of possible cheating that was used in this work.

2.4 Review of Relevant CNN Architectures

Deep learning models have the benefit of requiring very little feature engineering on data. They do not require features to be handcrafted nor even to be '*told*' which features they should learn on. Instead, these models learn the features from raw data themselves. It is, therefore, unnecessary to handcraft features for, say, VFOA or head pose data then train the model on those specific features. It suffices to train a deep learning model on well-labelled data. For this reason, this study only considered deep learning and, specifically, Convolutional Neural Network architectures for the task of detecting cheating.

CNN are one of the best machine learning algorithms for computer vision tasks (Khan et al., 2020). They have achieved greater levels of accuracy than traditional machine learning algorithms in fields such as image classification, image segmentation and object detection (Khan et al., 2020; Mahony et al., 2020).

However, there are several variations of CNN architectures; all with varying properties such as speed, accuracy, and input types, which affect where and how their derived models should be deployed.

This section reviews some of the CNN architectures that may be suitable for detecting possible cases of cheating using images from a webcam.

2.4.1 Single-frame 2D CNN

The typical single-frame 2D CNN has been used successfully in tasks such as eye gaze detection (Meng and Zhao, 2017), body pose estimation (Toshev and Szegedy, 2014) and other tasks somewhat related to detecting cheating. They have shown relatively high levels of accuracy and speed.

Another benefit of this architecture is that, because of its popularity, there are several open-source tools and libraries available that provide things such as out-of-the-box input pipelines, data pre-processing tools, and pre-trained state-of-the-art models.

This architecture's main drawback with regard to detecting cheating is that it completely ignores temporal data (Carreira and Zisserman, 2018). It, therefore, cannot fully interpret actions because actions have a temporal dimension. This is a drawback because cheating is an action and can therefore be better (but not exclusively) be understood by a model that can understand actions.

2.4.2 CNN + LSTM

This architecture, popularized by Donahue et al. (2016), adds a recurrent layer (usually LSTM) on top of a CNN. The CNN forms representations on an image (in the spatial domain), then the recurrent layer, using a sequence of output from the CNN segment, forms representations in the time domain.

Donahue et al. (2016) found that this architecture afforded a slightly higher accuracy of 68.20% over the typical 2D CNN architecture which achieved 67.37% accuracy on the UCF101 human action dataset (Soomro et al., 2012).

In another work, Carreira and Zisserman (2018) were able to achieve an accuracy of 91.0% on the UCF101 dataset using this architecture.

This architecture can capture temporal data because of the added recurrent layer. It can therefore understand actions to some extent. Also, concerning its CNN segment, it shares some of the benefits of 2D CNNs mentioned in the preceding sub-section.

However, the addition of the recurrent layer adds parameters, which makes it more computationally expensive and slower. Secondly, this architecture requires more data to train. This is because for it to understand actions, it needs to be trained on action data. And, for action data, a sample is a sequence of T images (where T has been observed to be, usually, above 7 (Carreira and Zisserman, 2018; Donahue et al., 2016)). Therefore, what would be X samples for a typical 2D CNN, would be X/T samples for this architecture.

2.4.3 3D CNN

The 3D CNN, introduced by Tran et al. (2015), extends the 2D CNN by adding a temporal dimension to the CNN's filters to make them 3-dimensional. This enables it to work on both spatial and temporal dimensions. Like the CNN+LSTM architecture, this architecture uses a sequence of images as a sample.

Because of the added dimension, this architecture usually has significantly more parameters compared to its 2D counterpart. This makes it prone to overfitting and largely limits the benefits, in terms of accuracy, it can derive from increasing depth. It also leads to high computational cost and, thus, low speeds (Carreira and Zisserman, 2018).

Also, like the CNN+LSTM architecture, since this architecture also takes a sequence of images as a single sample, it requires more data to train on than the typical 2D CNN.

2.5 Related Work

This section reviews related systems. The reviews are, however, limited to visual monitoring of candidates' activities during an exam, per the scope of this work.

2.5.1 Automated Online Exam Proctor

The Automated Online Exam Proctor (Atoum et al., 2017) is a fully automated online exam invigilation system. With regard to visual monitoring, this system uses two cameras – a webcam (figure 2-1, right) and another camera, dubbed wearcam, fixed on spectacles (figure 2-1, left). Using the input from these two cameras, the system performs the following invigilation tasks.

First, it performs text detection to determine whether a candidate might be cheating using a book. Secondly, it performs gaze estimation to determine the approximate direction where a candidate is looking. And lastly, it performs phone detection to determine whether a candidate may be using a phone or a similar device.

Figure 2-1: Automated Online Exams Proctor camera setup (Atoum et al., 2017)

2.5.2 The Exam Proctor Robot

The Exam Proctor Robot (Rosen and Carr, 2013) is a small, inexpensive gadget (Figure 2-2). It contains a camera that can turn in both altitude and azimuth, thus providing a 360-degree coverage of a candidate's environment. It is also fitted with an array of acoustic sensors which provide 3-dimensional directional data of acoustic events, probably, to inform the system on where it should point the camera.

Figure 2-2: The Exam Proctor Robot (Rosen and Carr, 2013)

This system does not, however, automate the detection of cheating. It only provides visual access to a candidate's environment.

2.5.3 Massive Open Online Proctor

The Massive Open Online Proctor (Li et al., 2015) is a proposed solution for online invigilation at the massive scale of Massive Open Online Courses (MOOCs).

With regard to visual monitoring of the candidates' behaviour, this system uses two cameras and a gaze tracker as input devices (figure 2-3). The cameras (circled in red) capture video of a candidate and his environment, and their input is used to perform action recognition. The

gaze tracker (circled in green), on the other hand, captures eye gaze data which is analysed and classified as ‘cheating’ or ‘non-cheating’.

Figure 2-3: MOOP setup (Li et al., 2015)

2.6 Gaps in Existing Systems

All the mentioned systems take a similar approach to tackle the problem of a limited view of candidates in online invigilation. They all attempt to increase this view so that invigilators have a more extensive view of candidates' environments.

However, this approach requires these systems to use additional input devices besides the ubiquitous laptop webcam, for example, the additional cameras. This introduces problems such as increased cost and complexity, and reduced accessibility for some students.

An automated invigilation system that uses only the input devices that come with a device (e.g., a laptop webcam and microphone), without requiring students to acquire external and additional devices, would make online invigilation accessible to a larger demographic of students.

2.7 Conceptual Framework

The following is a conceptual framework of the proposed system.

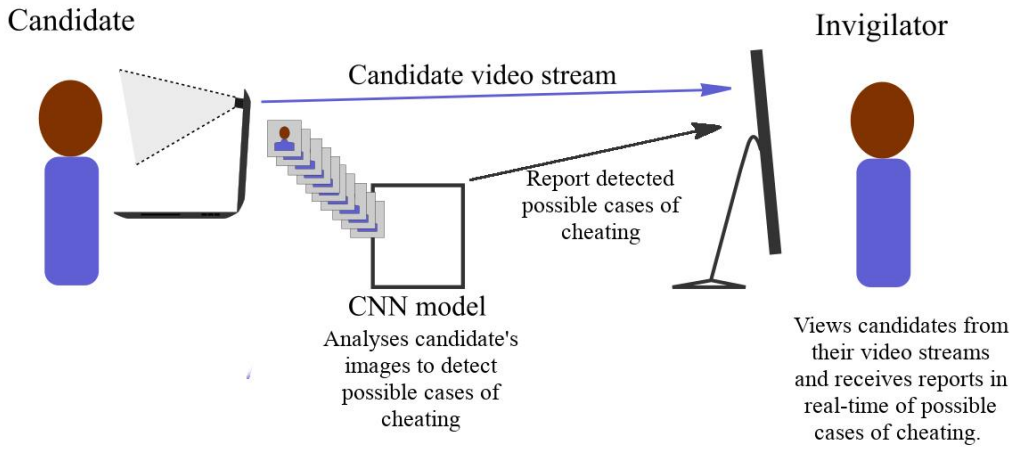

Figure 2-4: Conceptual framework

3 Research Methodology

3.1 Introduction

This section discusses the system development methodology used to develop the online invigilation application and the CNN model for detecting possible cases of cheating.

It should be noted that a distinction is made between the development methodologies for the online invigilation application and the CNN model.

3.2 System Development Methodology for the Online Invigilation Application

The online invigilation platform was developed using the Lean Software Development (LSD) methodology. It is an Agile methodology that is guided by seven principles; these are - *eliminate waste*, *amplify learning*, *decide as late as possible*, *deliver as fast as possible*, *empower the team*, *build integrity in* and *see the whole*. It was chosen due to its emphasis on the elimination of waste (i.e., anything that does not add value, in this case, towards fulfilling the research objectives) and fast delivery. These were important to speed up the development process and shorten the feedback loop.

On top of the LSD methodology, the Agile Software Development Life Cycle steps (figure 3-1) was used. These steps are discussed in the following subsection.

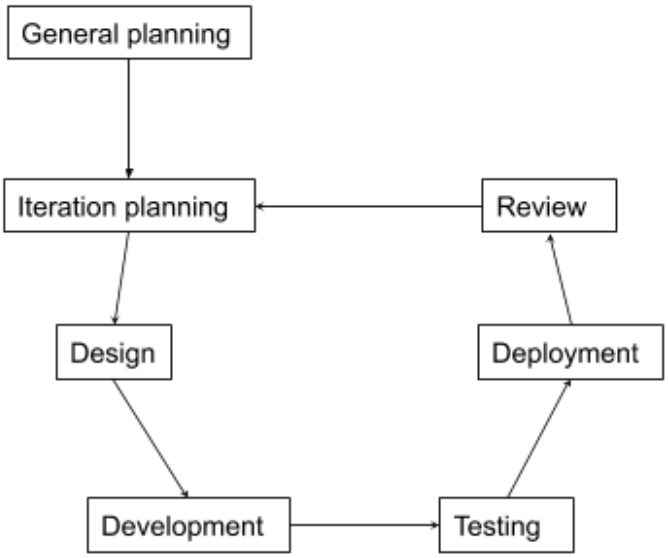

Figure 3-1: Agile Software Development Life Cycle

3.2.1 General Planning

This step involved the establishment of the basic requirements for the online invigilation platform.

It also involved an assessment of the project's feasibility by estimating the time and resources required to fulfil the requirements and the complexity involved.

3.2.2 Iteration Planning

This step started each iteration. It involved choosing a requirement among those established in the previous step (usually, the most important one) and planning on it more comprehensively, such as by breaking it up into finer elements and scheduling. The chosen requirement would be the task carried out throughout an iteration.

3.2.3 Design

At this step, conceptual designs for an iteration's requirement were developed using software architecture diagrams or mock-ups for user interface elements. Also, the tools, for example, libraries and frameworks, for implementing the iteration's requirement were identified at this step.

3.2.4 Development

This step involved the actualization of the designs developed in the previous step by programming and using the tools identified in the previous step.

3.2.5 Testing

At this step, the developed software was assessed to determine whether it fulfilled the iteration's requirement.

Also, the code added to the system was assessed to ensure it was of good quality and to avoid the accumulation of technical debt.

3.2.6 Deployment/Delivery

At this step, the developed and tested software or feature was integrated into the rest of the system and deployed onto a platform from which it could be interacted with and reviewed.

3.2.7 Review and Feedback

This step marked the end of an iteration. It would involve an assessment of the progress towards the fulfilment of all the project's requirements. Also, an assessment of the iteration's success

would be done, factoring in the reviews of external parties like the project's supervisor when it was given. The assessment performed at this stage would determine whether to start a new iteration.

3.3 Development Methodology of the CNN Model

The CNN model for detecting possible cases of cheating was developed using the following methodology.

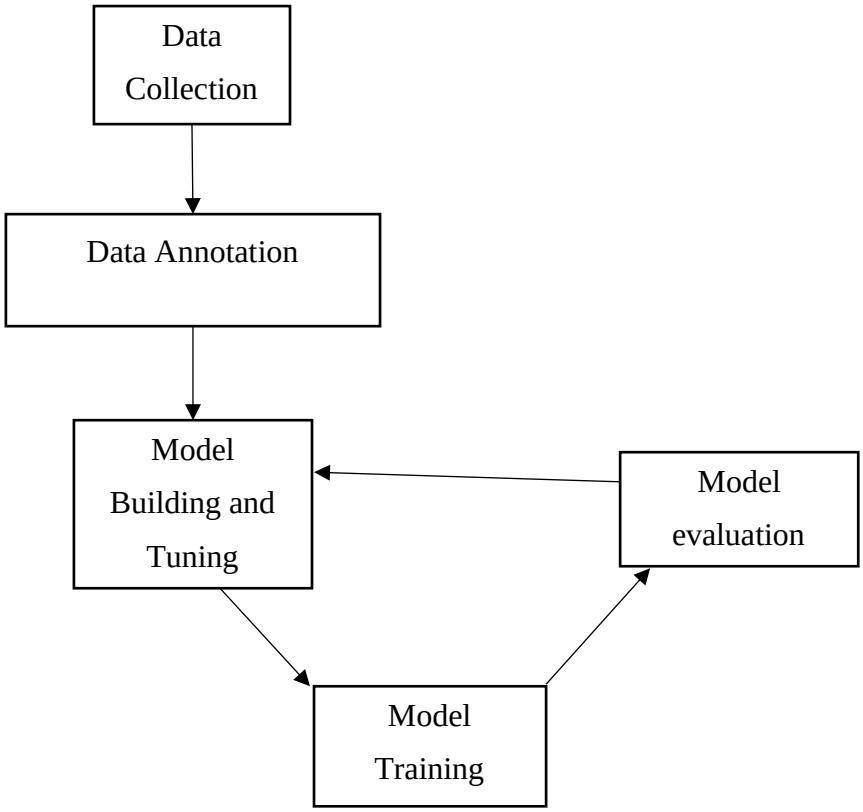

Figure 3-2: CNN Model development steps

The steps involved are as follows.

3.3.1 Data Collection

This step involved searching for available secondary data of good quality and collecting primary data.

3.3.2 Data Annotation

Here the collected data, which would be in video form, would be broken down into frames/images. Then each frame would be annotated as either regular or irregular. A regular frame was one that showed no cheating, whereas an irregular frame was one that captured a candidate cheating.

3.3.3 Model Building and Tuning

This step involved the assembly or tuning of a Convolutional Neural Network model.

3.3.4 Model Training

Here, the built or tuned model would be trained on a portion of the prepared dataset.

3.3.5 Model Evaluation

At this step, the trained model would be tested using the remaining portion of prepared data. The test results would then be compared to previous results.

After this step, the process would restart from the model building or tuning step until satisfactory test results were obtained.

3.4 System Development Tools and Technologies

3.4.1 React

React is a JavaScript framework for building user interfaces. It enables one to write maintainable and reusable user interface code easily and quickly. It also allows for easy integration with external tools. For these reasons, it was found to be a suitable choice for developing the online invigilation system's user interface.

3.4.2 Laravel

Laravel is a web application development framework that also provides tools and structures for building backend systems. It provides many out-of-the-box features and tools and is easy to use. Laravel was used to develop the system's backend; to perform things such as authentication and coordination of real-time communication between users' devices.

3.4.3 WebRTC

WebRTC is an open framework that allows real-time, peer-to-peer communication of data, including video, in a browser. It was used to develop the video streaming functionality that would allow invigilators to view candidates' live, as they take an exam.

3.4.4 Pusher

Pusher is a service that provides an Application Programming Interface (API) and the necessary infrastructure for sending real-time messages between devices. It was primarily used in establishing WebRTC connections.

3.4.5 TensorFlow, Tensorflow.js and Keras

TensorFlow is an open-source machine learning platform that provides a plethora of tools and features that make it easy and fast to train, test and deploy machine learning models.

Tensorflow.js is a JavaScript variant of TensorFlow. It allows deployment of models in a browser. It was used to deploy the CNN cheating detection model.

Keras is a deep learning API running on top of TensorFlow that provides abstractions for deep learning models and tools for manipulating these models. It also provides an array of state-of-the-art pretrained models without cost. It was used in the construction of the CNN cheating detection model.

3.5 System Deliverables

3.5.1 CNN Cheating Detection Model

The CNN cheating detection model was the main contribution of this work. By analysing candidates' images, it detects whether a candidate might be cheating. It was required to be fast so that it could analyse images in real-time. It also had to be highly accurate to reduce false positives and, more importantly, false negatives.

3.5.2 Candidate's Module

The candidate's module captures and streams a candidate's video to an invigilator's device, thus allowing invigilators to monitor candidates live. It also uses the CNN model to detect and report when a candidate may be cheating.

3.5.3 Invigilator's Module

The invigilator's module receives the video stream from the candidate's module and displays it to the invigilator. It also receives alerts from the candidate's module when the CNN model has detected a possible cheating case.

4 System Analysis and Design

4.1 Introduction

This chapter contains the system requirements and the system analysis and design diagrams of the online invigilation system.

4.2 System Requirements Analysis

The main requirements of the online invigilation system are the following.

4.2.1 Functional Requirements

4.2.1.1 Detecting and Reporting Possible Cases of Cheating

The system can detect possible cases of cheating that are within the scope mentioned in section 1.7 of this document. This is done using the developed CNN model and image data taken from candidates' webcams.

Further, whenever the system detects a possible incident of cheating, it alerts the invigilator in real-time.

4.2.1.2 Streaming Candidates' Video to Invigilators

The online invigilation system can stream candidates' video to invigilators' devices, for candidates and invigilators in the same exam session. This enables invigilators to view candidates live as they are taking an exam. This is achieved by using the WebRTC framework to establish peer-to-peer connections between the candidates and invigilators and then using those connections to stream video taken from the candidates' webcams to the invigilator's module.

4.2.2 Non-functional Requirements

4.2.2.1 Realtime

The system works in real-time. First, the CNN model analyses candidates' images in real-time. Second, invigilators can monitor candidates in real-time from their webcam video. And lastly, the system reports detected possible cases of cheating in real-time. This allows invigilators to act accordingly as soon as it is detected that a candidate may be cheating.

4.3 System Analysis and Design Diagrams

The following are the system analysis and design diagrams that were used.

4.3.1 Use Case Diagram

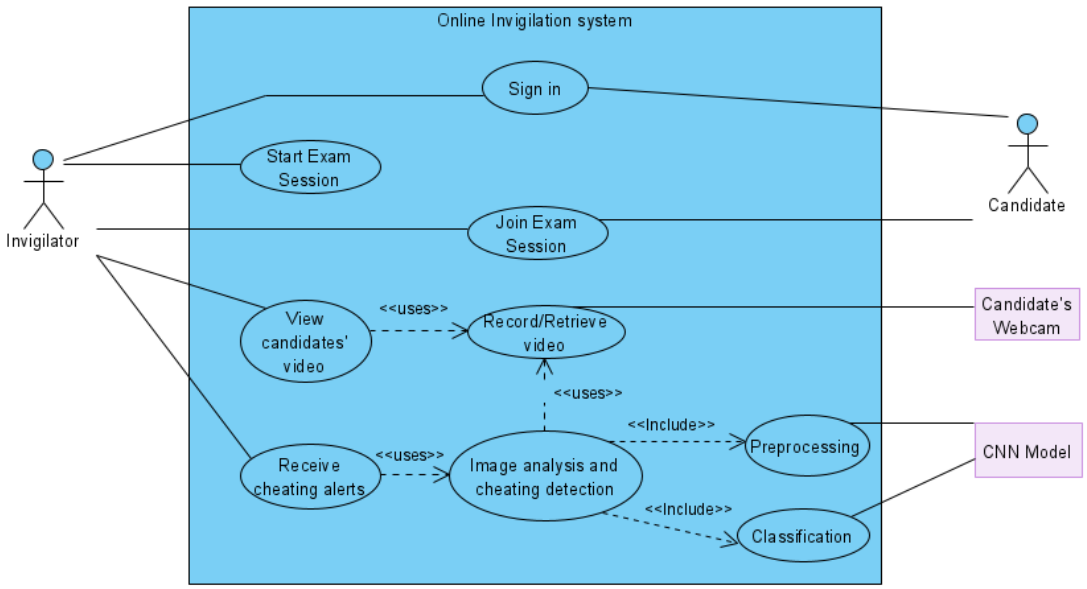

Figure 4-1: Use case diagram

The use case diagram shows the actions invigilators and candidates can perform on the online invigilation system. An invigilator can sign in, start an exam session, join the exam session, monitor candidates as they take an exam and receive alerts of possible cheating events, which are automatically detected by the system. A candidate can sign in and join an exam session.

4.3.2 Sequence Diagram

The sequence diagram below shows the sequence of interactions that can happen between the various modules and subsystems of the online invigilation system.

First, an invigilator must create an exam session, after which they will get a unique code for the exam session. Using the code, the invigilator himself and candidates can join the exam session. When a candidate joins an exam session, the invigilator's module is alerted. Once alerted, it initiates the establishment of a peer connection with the candidate's module, from which it will get a candidate's video stream. After this, while the exam is still in session, the candidate's module will be continually capturing images of the candidate and using the CNN cheating detection model to analyse them for possible cases of cheating. If such a case is detected, then the candidate's module will send an alert to the invigilator's module.

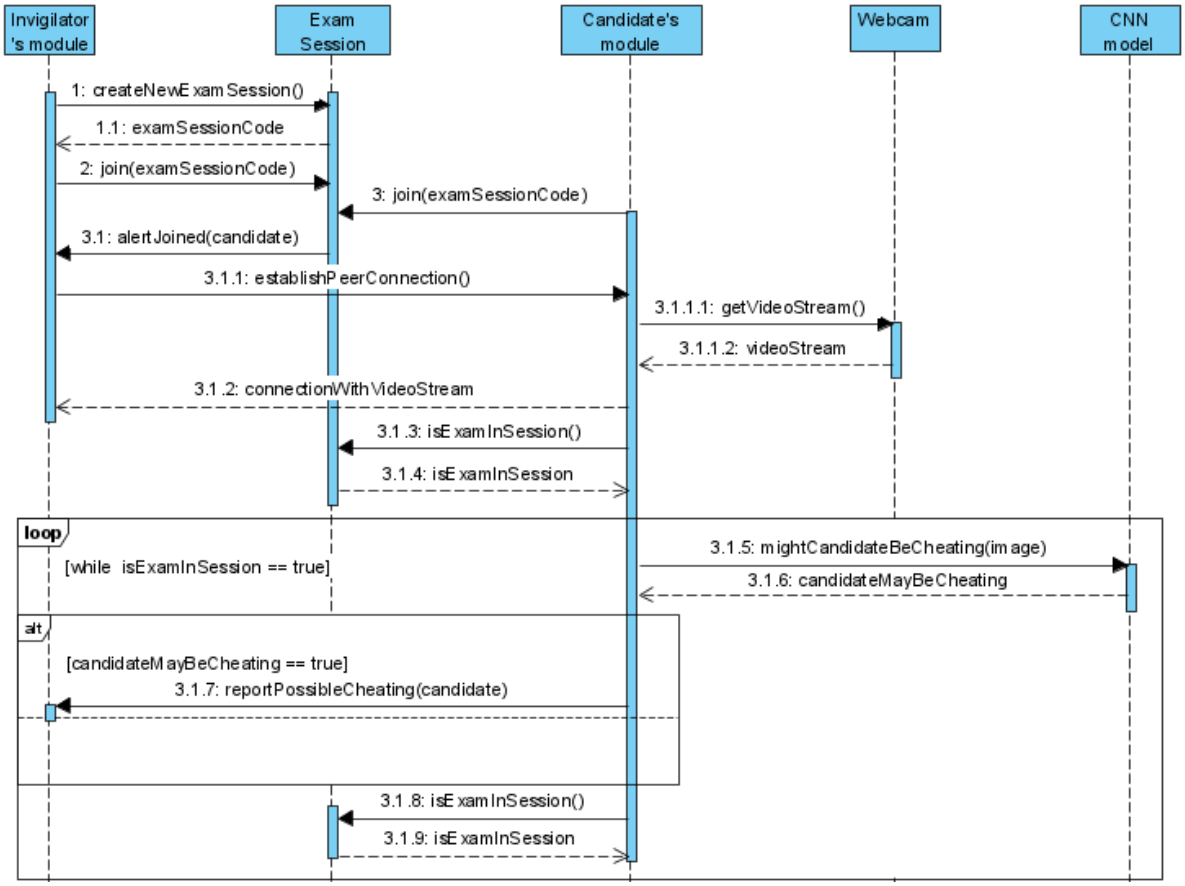

Figure 4-2: Sequence Diagram

4.3.3 Database Schema

The system did not extensively use a database. The only uses of a database were to store invigilators' and candidates' credentials, and exam sessions' data. The following is a database schema representing the database structure of the online invigilation system.

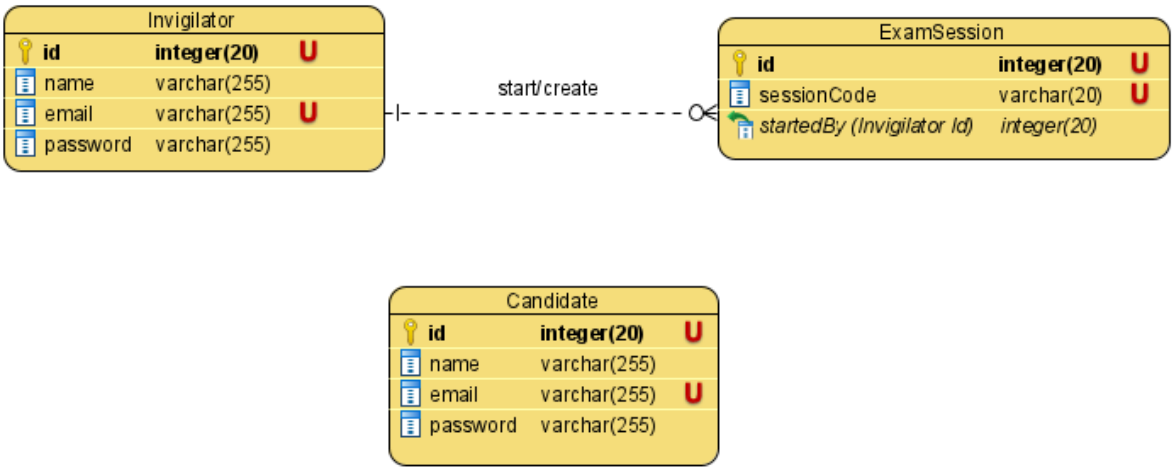

Figure 4-3: Database Schema

5 System Implementation and Testing

5.1 Introduction

This section describes the implementation and testing details of the CNN cheating detection model and the online invigilation application.

5.2 Implementation Environment

5.2.1 Hardware Specifications

The development of the CNN cheating detection model was done on T4 and P100 GPUs, about 12 GB of RAM and about 145 GB of disk space, as provided by Google's Colab Pro.

The online invigilation application was developed and run on a device with a dual-core 2.0 GHz CPU, about 8 GB of RAM and about 512 GB of disk space.

5.2.2 Software Specifications

As mentioned before, the CNN model was developed on Google's Colab Pro application using TensorFlow and Keras. TensorFlow.js was used in deploying the model.

The online invigilation application was developed on a device running Ubuntu 18.04 OS and was run and tested on Google's Chrome browser.

5.3 CNN Cheating Detection Model Implementation and Testing

5.3.1 Dataset Preparation and Description

This section describes the process followed in obtaining and annotating the data used to train and test the CNN model. It also describes the data itself.

5.3.1.1 Data Collection

Because of scarcity and inaccessibility of good quality secondary data, primary data was collected. A simple web application was developed and used in the collection process (see Appendix). Participants took a mock exam on the application using laptops with a webcam. To encourage cheating, the participants were asked to open a link to a website with the exam answers on their phones. Also, the questions were made intentionally difficult.

Throughout the mock exam, the participants' video was being recorded and stored by the application.

14 participants participated in the data collection process, providing 19 videos. However, 5 of the videos were excluded because they did not meet the required standard of quality for reasons such as insufficient illumination or participants' failure to follow the given instructions. The 14 videos that met the quality requirements were split into two sets – the training and test set. The training set contained 9 videos while the test set contained 5. Of the 5 videos that constituted the test set, 3 were of participants whose videos were not also in the training set.

5.3.1.2 Data Annotation

The videos were broken down into frames/images with an appropriate frame rate. Then, using the Supervisely annotation software, each frame was labelled as either regular (no sign of cheating) or irregular (signs of cheating present). Signs of cheating were when a participant's direction of visual attention was off their laptop's screen or keyboard and (probably) on their phone, to the left or right of their laptop or under their desk (figure 5-1, right).

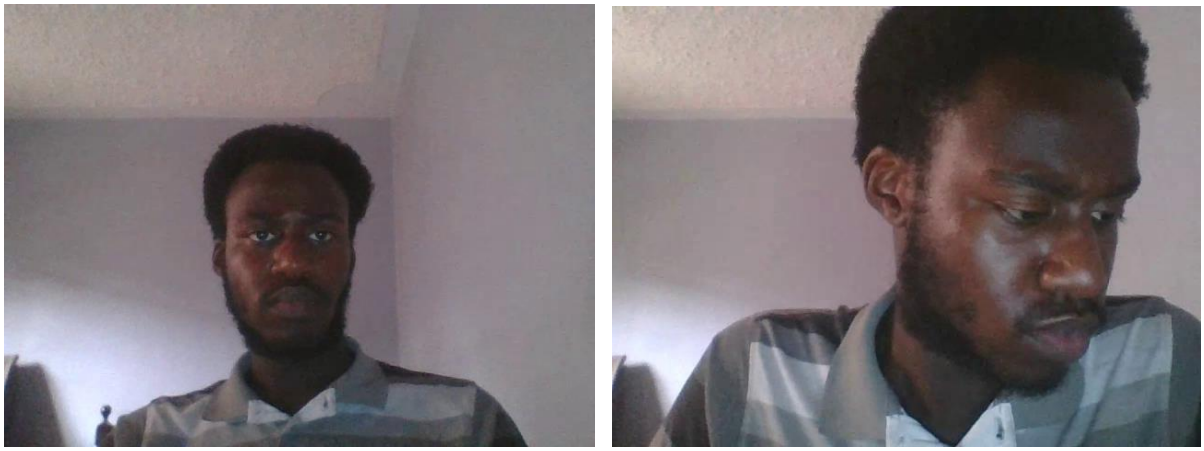

Figure 5-1: Left: A frame annotated as regular since the participant's direction of visual attention is on their screen. Right: A frame annotated as irregular since the participant's direction of visual attention is off their screen and to the left of their laptop.

When this step was completed, the training set had 2890 and 2048 frames annotated as regular and irregular, respectively. The test set had 604 and 845 frames annotated as regular and irregular, respectively. The resulting complete dataset was dubbed the *online exam cheating* (OEC) dataset.

Several images were excluded from the OEC dataset for being near-duplicates of others or for poor image quality, among other things.

5.3.2 Model Training

5.3.2.1 Pre-Processing

All frames underwent the following pre-processing operations. First, they were resized to 150 by 150 pixel dimensions. Second, their pixel range was normalized to between 0 and 1. And lastly, because of the small size of the training dataset, data augmentation was used extensively to minimize overfitting. The data augmentation operations that were carried out were random horizontal flipping, zooming, translation, brightness variation, contrast variation and saturation variation.

The three pre-processing operations – resizing, normalization and data augmentation – were made a part of the cheating detection model to improve the model's portability and to reduce the complexity of deployment. However, the data augmentation operation was set to be active only during training.

5.3.2.2 Model Architecture

The CNN cheating detection model was built with a pretrained model as its base model so that it would benefit from transfer learning (Zhuang et al., 2020). Transfer learning was crucial due to the small size of the OEC dataset. The chosen pretrained model was a MobileNetV2 (Sandler et al., 2019) because of its high speed performance and relatively low memory and storage requirements.

A global average pooling layer was placed on top of the base model to downsample the base model's output. This approach was chosen instead of the more classic approach of using densely connected layers because, unlike the latter, it does not introduce new parameters, which may reduce the speed of the model and cause overfitting.

The model's last layer was a densely connected layer with one neuron, which used the sigmoid activation function to output the probability that a frame belonged to either the regular or irregular classes.

The complete CNN cheating detection model architecture is shown in figure 5-2.

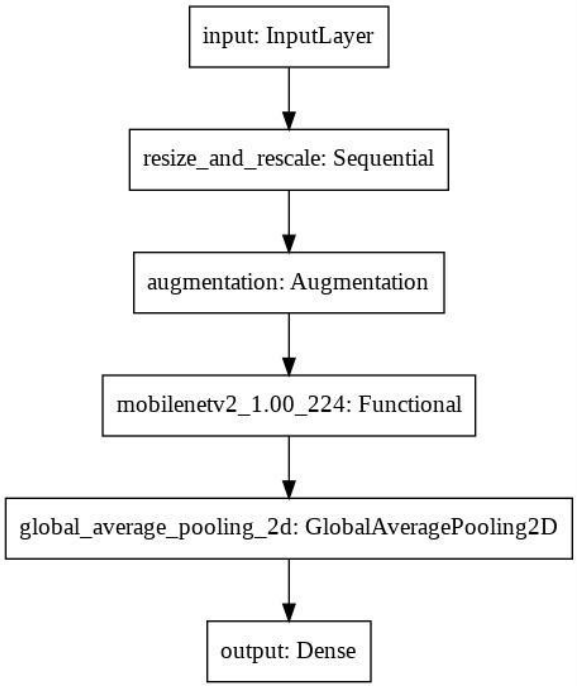

Figure 5-2: CNN cheating detection model

5.3.2.3 Training Process

The training of the model followed the following process. First, the pretrained base model (MobileNetV2) was frozen to prevent its weights from changing and thus maintain the information it contained. Then the cheating detection model was trained until it converged - so that it would learn to classify frames using the features extracted by the base model.

After this, the model was fine-tuned. Fine-tuning involved unfreezing the base model then re-training the entire model on the same dataset but with a very small learning rate. This was done to adapt the pretrained base model to the new dataset.

5.3.3 Results and Discussion

The CNN cheating detection model, when evaluated on the mentioned test set, yielded an accuracy of 79.92%, a precision of 74.79% and a recall of 89.01%.

Other significant metrics that were looked at were the model's parameter count and storage size. The parameter count was significant because it is a determinant of a model's speed, which was itself important in this work since the model was meant to be run in real-time and therefore had to be fast. Storage size was significant because of the approach taken for deploying the model as is discussed in section 5.3.4.

Table 1 shows the metrics of the cheating detection model when various top-grade pretrained models were used as its base model. MobileNetV2 yielded a relatively decent accuracy and

had the least parameter count and storage size. The best performing model, ResNet50V2, had 10 times the parameter count and storage size of MobileNetV2.

Base model	**Accuracy**	**No. of Parameters**	**Size (in MBs)[1]**
MobileNetV2	**79.92**	**2,257,984**	**8.63**
MobileNet	80.54	3,229,889	11.66
Xception	75.98	20,863,529	74.08
ResNet50V2	86.13	23,564,800	83.93
ResNet101V2	82.68[2]	42,628,609	450.75
InceptionV3	74.19	21,804,833	77.71

Table 1: Comparison of top-grade pretrained models on the cheating dataset. [1]Size was measured as the storage space of the files produced when the models were serialized. [2]The accuracy for ResNet101V2, unlike the others, was a mean of three accuracy results due to their high disparity.

Figure 5-3 shows an example of the cheating detection model's output. The probability that a participant is cheating increases when their head and eyes are turned to the left or right of their device, or under their desk. It is lowest when their head and eyes are facing their device.

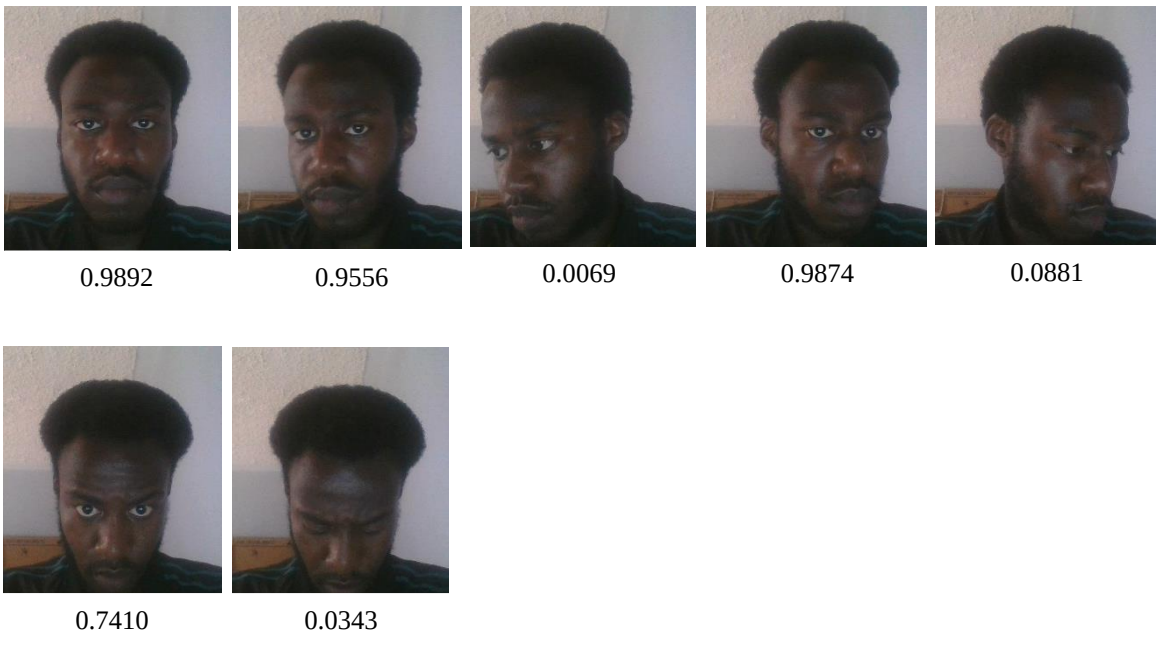

Figure 5-3: Example of model prediction output. The closer to 0.0 the prediction is, the more likely the participant is cheating.

5.3.4 Model Deployment

For deployment, the trained and tested model was serialized using TensorFlow. Then, using TensorFlow.js tools, the serialized model was converted to a TensorFlow.js-compatible format. After that, the serialized model was placed in an online server, where it would be fetched by a browser and deserialized using TensorFlow.js into a working model.

5.4 Online Invigilation Web Application Implementation and Testing

5.4.1 Implementation

5.4.1.1 Candidate's Module

The candidate's module allows candidates to join an exam session. To do so, a candidate requires an exam session's unique code which is generated when an invigilator creates an exam session.

After a candidate joins an exam session, in the background, this module uses Pusher's real-time messaging functionality to establish WebRTC peer connections with newly joining invigilators' devices. The module then uses established WebRTC peer connections to stream candidates' video to invigilators' devices for live monitoring.

The candidate's module is also responsible for loading and running the CNN cheating detection model. This involves fetching the serialized model and weight files from an online server, deserializing the files to form a working cheating detection model and providing the model with input, i.e., a candidate's images. These functions are carried out using TensorFlow.js.

Lastly, the module reports the CNN model's output to present invigilator's devices using the established WebRTC peer connections.

5.4.1.2 Invigilator's Module

The invigilator's module allows invigilators to create exam sessions or join on-going exam sessions. When an invigilator creates an exam session, he gets back a unique code for the new exam session, which candidates and other invigilators must use to join the exam session.

After joining an exam session, this module, like the candidate's module, uses Pusher's real-time functionality to establish WebRTC peer connections with newly joining candidate's devices. Further, this module uses established peer connections to retrieve candidates' live video streams and display them to an invigilator in an organized manner as shown in figure 5-4.

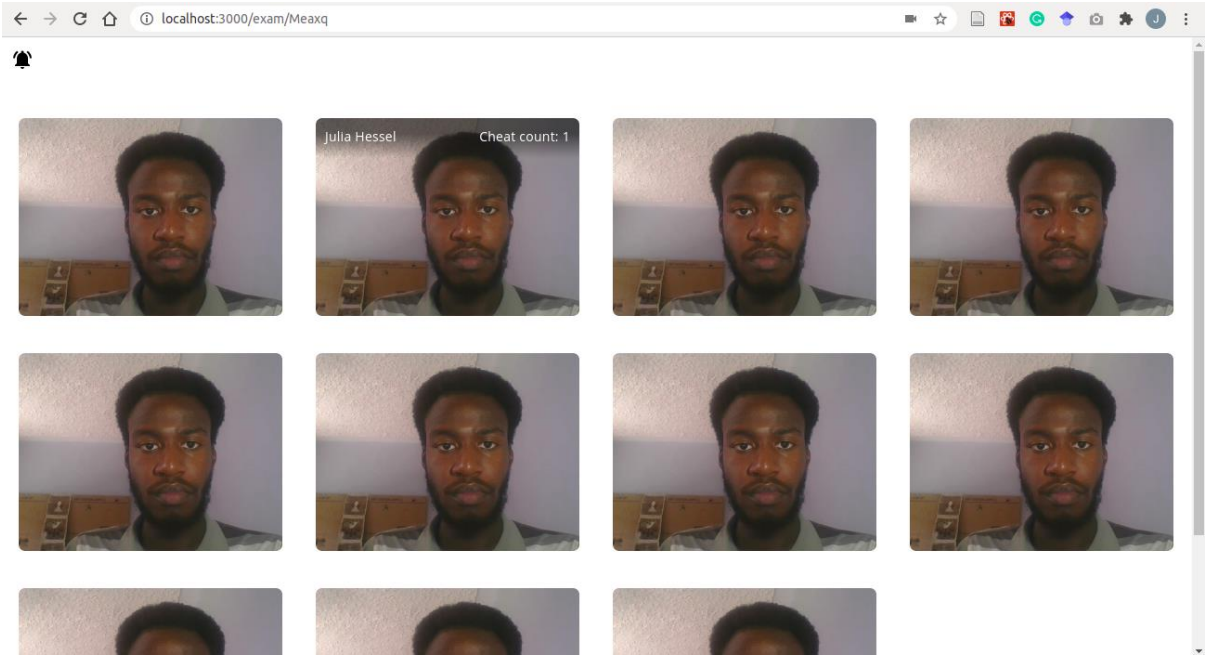

Figure 5-4: Invigilator's module 'exam-room' page, where invigilator can view monitor candidates live.

When a possible case of cheating is reported from the candidate's module, this module receives the report and displays an alert to an invigilator in a noticeable way (figure 5-5). Further, it arranges candidate's video streams so that the video streams of candidates with the highest reports of cheating appear at the top.

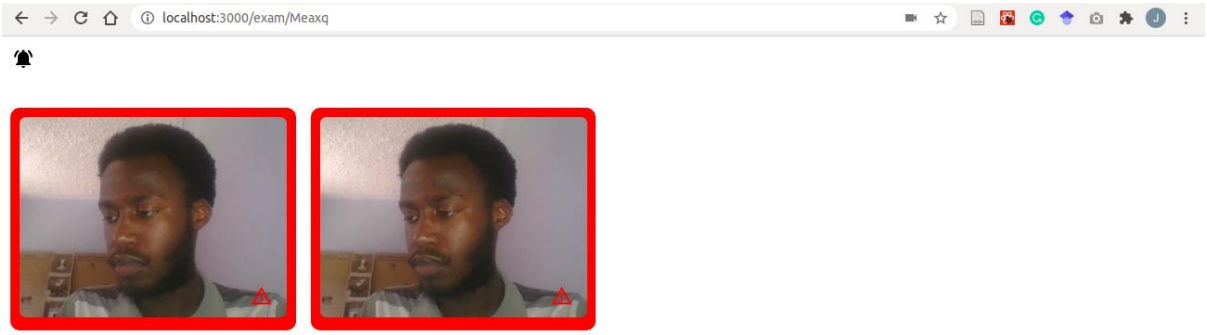

Figure 5-5: Invigilator's module displaying report of possible incidents of cheating

5.4.2 Testing

The implemented application was tested to verify that it meets its requirements. Black box testing was used. Tests were run manually.

5.4.2.1 Candidate's Module Test Cases

Test Case	**Description**	**Input**	**Result**	**Test Verdict**
Providing invalid exam session code	When an invalid exam session code is provided, the system should display an error message to the user saying the code is invalid.	Invalid exam session code	The system displays an error message to inform the user that the provided exam session code is invalid	Passed
Providing valid exam session code	When a user provides a valid exam session code, they should be redirected to the exam session to which the code belongs.	Valid exam session code	The user is redirected to the appropriate exam session	Passed
Candidate's video streamed to invigilators	When candidates join an exam session, the video from their webcams should be streamed to present invigilator's devices.	Manually joining an exam session having an invigilator present	The video from the candidate's webcam is streamed to an invigilator's device	Passed
Reports detected cases of cheating to an invigilator	When a candidate performs cheating activities that are within the specified	Image input of candidate performing cheating	Reports are sent to invigilator's devices	Passed

	scope, a report should be sent to an invigilator's device.	activities taken from his webcam.		

5.4.2.2 Invigilator's Module Test Cases

Test Case	**Description**	**Input**	**Result**	**Verdict**
Creating an exam session	When an invigilator creates an exam session, the system should return the unique code for the new exam session.	Manually create an exam session from the user interface	The code for the exam session is displayed to the invigilator.	Passed
Providing invalid exam session code	When an invalid exam session code is provided, the system should display an error message to the user saying the code is invalid.	Invalid exam session code	The system displays an error message to inform the user that the provided exam session code is invalid	Passed
Providing valid exam session code	When a user provides a valid exam session code, they should be redirected to the appropriate exam session page.	Valid exam session code	The user is redirected to the appropriate exam session	Passed
Receives and displays candidates' video streams	When both invigilators and candidates are present in an exam session, the candidates' live video should be	Manually having both a candidate and an invigilator joining an exam session at the same time.	Candidate's video is played on the invigilator's device	Passed

	played on the invigilators' devices.			
Alerts are produced when cheating is detected	When cheating is detected, an alert should be emitted, showing the specific candidate who may be cheating.	Image input of candidate performing cheating activities.	Alerts are produced, showing the candidate who may be cheating	Passed

6 Conclusion, Recommendations, and Future Works

6.1 Conclusion

This work sought to reduce the effort required in the invigilation of online exams by developing an automated invigilator's aid. To the extent of this work's scope, as specified in this document, the researcher is of the view that this has been achieved. The developed system is able to automate the detection of possible cheating incidents as indicated by candidates' direction of visual attention and to report such cases. However, it is also true that the developed system is not a comprehensive solution. There are many features that may be added to the system to enhance its effectiveness, efficiency and readiness for use in the wild.

6.2 Recommendations

It is recommended that more extensive data collection should be carried out so that a bigger and more diverse dataset is obtained for training and testing the cheating detection model. Further, if possible, the data should be collected in an environment and using a procedure that, as closely as possible, resemble an actual exam. This will ensure that the data reflects the real world as much as possible.

It is also recommended that, if a bigger and more diverse dataset is available, more indicators of cheating should be considered. Also, the cheating detection model should be developed to classify, separately, each cheating activity, so that invigilators will additionally be alerted on what specific activity a purportedly cheating candidate may be performing.

6.3 Future Works

Future research should attempt to develop cheating detection models that can interpret video data and not just image data. The CNN + LSTM model and 3D CNN architectures mentioned in sections 2.4.2 and 2.4.3 are valid candidates for this. This will make the invigilation system more efficient, especially by reducing false positives. However, these models should be developed in view of their lower speeds and higher data requirements as compared to the model architecture used in this work.

7 References

Allen, I.E., Seaman, J., 2007. Online Nation: Five Years of Growth in Online Learning 31.

Asep, H.S.G., Bandung, Y., 2019. A Design of Continuous User Verification for Online Exam Proctoring on M-Learning, in: 2019 International Conference on Electrical Engineering and Informatics (ICEEI). Presented at the 2019 International Conference on Electrical Engineering and Informatics (ICEEI), pp. 284–289. https://doi.org/10.1109/ICEEI47359.2019.8988786

Atoum, Y., Chen, L., Liu, A.X., Hsu, S.D.H., Liu, X., 2017. Automated Online Exam Proctoring. IEEE Trans. Multimed. 19, 1609–1624.

Ba, S.O., Odobez, J., 2009. Recognizing Visual Focus of Attention From Head Pose in Natural Meetings. IEEE Trans. Syst. Man Cybern. Part B Cybern. 39, 16–33.

Balbuena, S.E., Lamela, R.A., 2015. Prevalence, Motives, and Views of Academic Dishonesty in Higher Education, Online Submission.

Berry, R., 2008. Introduction, in: Assessment for Learning. Hong Kong University Press, pp. 1–4.

Carreira, J., Zisserman, A., 2018. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset.

Chuang, C., Femiani, J., Craig, S., 2015. The Role of Certainty and Time Delay in Students' Cheating Decisions during Online Testing.

Chuang, C.Y., Craig, S.D., Femiani, J., 2017. Detecting probable cheating during online assessments based on time delay and head pose. High. Educ. Res. Dev. 36, 1123–1137. https://doi.org/10.1080/07294360.2017.1303456

Crown, D.F., Spiller, M.S., 1998. Learning from the Literature on Collegiate Cheating: A Review of Empirical Research. J. Bus. Ethics 17, 683–700. https://doi.org/10.1023/A:1017903001888

Donahue, J., Hendricks, L.A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., Darrell, T., 2016. Long-term Recurrent Convolutional Networks for Visual Recognition and Description.

Freiburger, T.L., Romain, D.M., Randol, B.M., Marcum, C.D., 2017. Cheating Behaviors among Undergraduate College Students: Results from a Factorial Survey. J. Crim. Justice Educ. 28, 222–247. https://doi.org/10.1080/10511253.2016.1203010

Kerkvliet, J., Sigmund, C.L., 1999. Can We Control Cheating in the Classroom? J. Econ. Educ. 30, 331–343. https://doi.org/10.2307/1182947

Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S., 2020. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Artif. Intell. Rev. https://doi.org/10.1007/s10462-020-09825-6

Khan, H., Williams, J., 2006. Poverty Alleviation Through Access to Education: Can E-Learning Deliver? SSRN Electron. J. https://doi.org/10.2139/ssrn.1606102

Li, X., Chang, K., Yuan, Y., Hauptmann, A., 2015. Massive Open Online Proctor: Protecting the Credibility of MOOCs Certificates, in: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '15. Association for Computing Machinery, New York, NY, USA, pp. 1129–1137. https://doi.org/10.1145/2675133.2675245

Mahony, N.O., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco-Hernandez, G., Krpalkova, L., Riordan, D., Walsh, J., 2020. Deep Learning vs. Traditional Computer Vision. ArXiv191013796 Cs 943. https://doi.org/10.1007/978-3-030-17795-9

Mccabe, D., Trevino, L., Butterfield, K., 2001. Cheating in Academic Institutions: A Decade of Research. Ethics Behav. - ETHICS BEHAV 11. https://doi.org/10.1207/S15327019EB1103_2

McDowell, L., Sambell, K., 2014. Assessment for Learning Environments: A Student-Centred Perspective, in: Advances and Innovations in University Assessment and Feedback. Edinburgh University Press, pp. 56–72.

Meng, C., Zhao, X., 2017. Webcam-Based Eye Movement Analysis Using CNN. IEEE Access 5, 19581–19587. https://doi.org/10.1109/ACCESS.2017.2754299

Prince, D.J., Fulton, R.A., Garsombke, T.W., 2009. Comparisons Of Proctored Versus Non-Proctored Testing Strategies In Graduate Distance Education Curriculum. J. Coll. Teach. Learn. TLC 6. https://doi.org/10.19030/tlc.v6i7.1125

Rosen, W.A., Carr, M.E., 2013. An autonomous articulating desktop robot for proctoring remote online examinations, in: 2013 IEEE Frontiers in Education Conference (FIE). pp. 1935–1939. https://doi.org/10.1109/FIE.2013.6685172

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2019. MobileNetV2: Inverted Residuals and Linear Bottlenecks. ArXiv180104381 Cs.

Smith, K., Ba, S., Odobez, J.-M., Gatica-Perez, D., 2008. Tracking the Visual Focus of Attention for a Varying Number of Wandering People. IEEE Trans. Pattern Anal. Mach. Intell. 30, 1212–29. https://doi.org/10.1109/TPAMI.2007.70773

Software Secure, 2016. Eyes on Integrity A Comparative Look at Online Proctoring Models.

Soomro, K., Zamir, A.R., Shah, M., 2012. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild.

Steve Kolowich, 2013. Behind the Webcam's Watchful Eye, Online Proctoring Takes Hold [WWW Document]. Chron. High. Educ. URL https://www.chronicle.com/article/behind-the-webcams-watchful-eye-online-proctoring-takes-hold/ (accessed 2.1.21).

Toshev, A., Szegedy, C., 2014. DeepPose: Human Pose Estimation via Deep Neural Networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition. Presented at the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Columbus, OH, USA, pp. 1653–1660. https://doi.org/10.1109/CVPR.2014.214

Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. ArXiv14120767 Cs.

Woldeab, D., Lindsay, T., Brothen, T., 2017. Under the Watchful Eye of Online Proctoring, in: Innovative Learning and Teaching: Experiments Across the Disciplines.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., 2020. A Comprehensive Survey on Transfer Learning. ArXiv191102685 Cs Stat.

Appendix: Screenshots of the Data Collection Site

Figure 0-1: Data collection site intro page

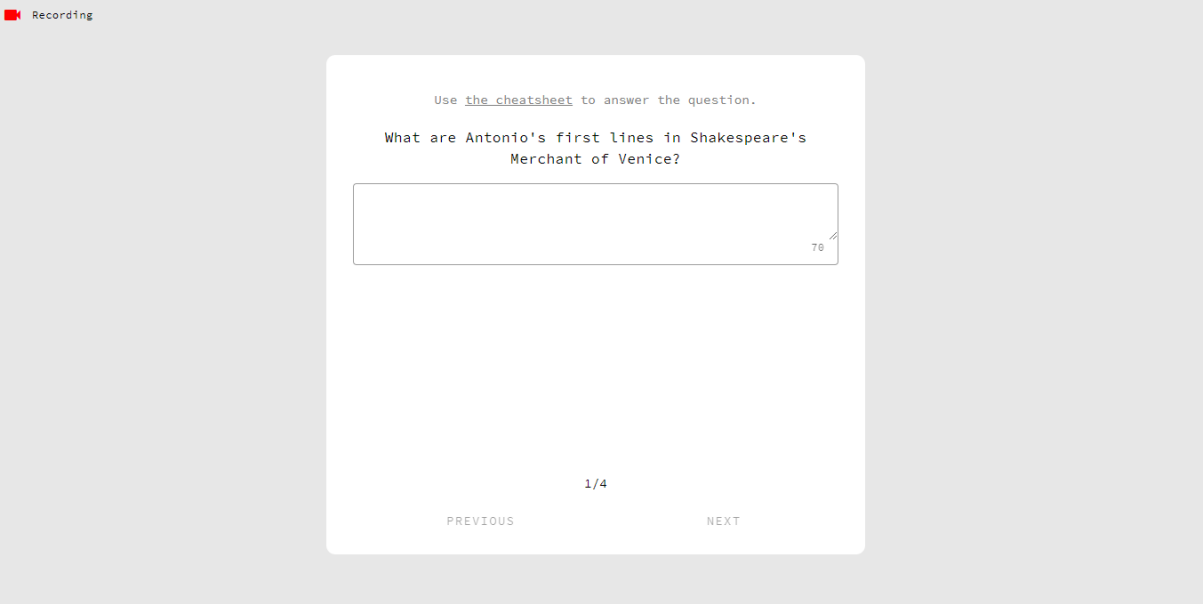

Figure 0-2: Data collection site test page

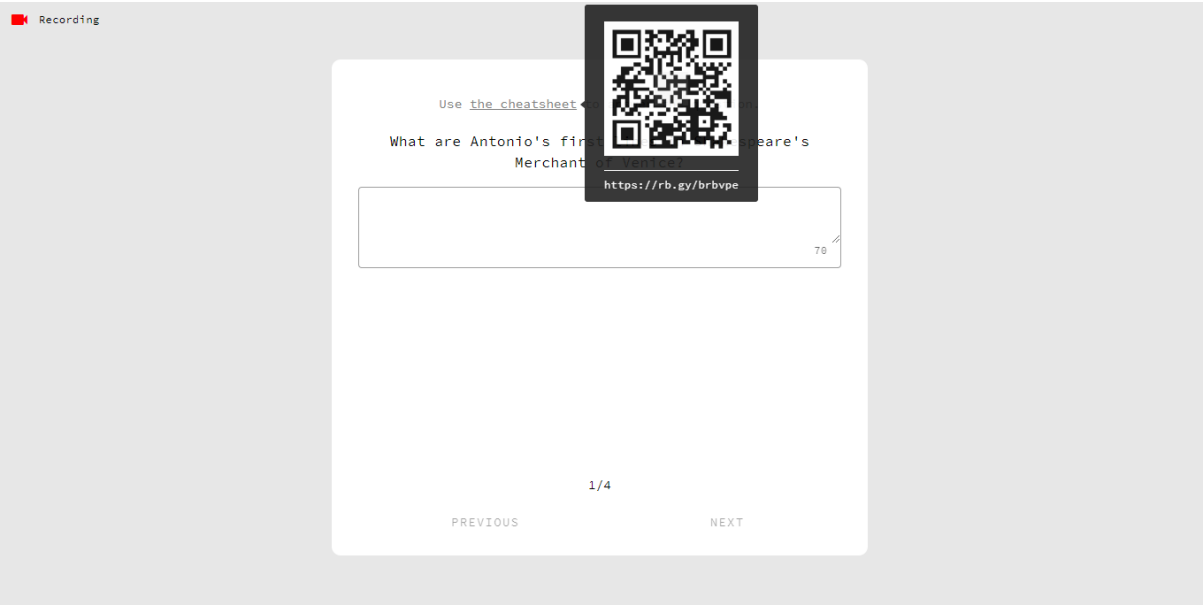

Figure 0-3: Data collection site test page with cheat-sheet link shown.