ISLAMIC UNIVERSITY OF TECHNOLOGY

Real time Gaze Tracking in Remote Proctoring: A Study of Appearance-based Gaze Estimation

Authors

Nafiul Onim (180042104)

Mirza Sadaf Shahid (180042137)

Muhammad Rafsan Quayes (180042141)

Supervised By

Dr. Hasan Mahmud
Associate Professor
Dept. of CSE, IUT

Fardin Saad
Lecturer
Dept. of CSE, IUT

Dr. Md. Kamrul Hasan
Professor
Dept. of CSE, IUT

Systems and Software Lab (SSL)
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
A Subsidiary Organ of the Organization of Islamic Cooperation.
Dhaka, Bangladesh.

A thesis submitted to the Department of CSE
in partial fulfilment of the requirements for the degree of B.Sc. in Software Engineering

Academic Year: 2021-2022

March 2023

Declaration of Authorship

Us, Nafiul Onim, Mirza Sadaf Shahid and Muhammad Rafsan Quayes , declare that this thesis titled, 'Real time Gaze Tracking in Remote Proctoring : A Study of Appearance-based Gaze Estimation' and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Any part of this thesis has not been submitted for any other degree or qualification at this University or any other institution.
- Where I have consulted the published work of others, this is always clearly attributed.

Submitted By: (Signature of the Candidates)

Nafiul Onim - 180042104

Mirza Sadaf Shahid - 180042137

Muhammad Rafsan Quayes - 180042141

Real time Gaze Tracking in Remote Proctoring: A Study of Appearance-based Gaze Estimation

Supervisors:

Dr. Hasan Mahmud
Associate Professor,
Department of Computer Science and Engineering,
Islamic University of Technology (IUT), Dhaka, Bangladesh.

Fardin Saad
Lecturer,
Department of Computer Science and Engineering,
Islamic University of Technology (IUT), Dhaka, Bangladesh.

Dr. Md. Kamrul Hasan
Professor,
Department of Computer Science and Engineering,
Islamic University of Technology (IUT), Dhaka, Bangladesh.

Abstract

Our thesis aims to address the critical issue of academic dishonesty in online examinations by proposing a proctoring system that integrates eye gaze tracking technology for the detection of suspicious behavior. The study begins by discussing the existing challenges of current examination systems and identifying the problems that need to be addressed. It emphasizes the necessity for a more advanced proctoring system with gaze tracking capabilities to effectively deter attempts at academic dishonesty. The research is divided into two main parts: the selection of an appropriate model and the incorporation of proctoring functionalities. Two models were chosen for evaluation, namely iTracker, which was pre-trained on the GazeCapture dataset, and L2cs-net, which we trained on the MPIIFaceGaze dataset. The findings from these experiments indicate that L2cs-net outperforms iTracker in terms of accuracy, speed, and latency but only when supplied with the processing power of a GPU, without one iTracker is better. Regarding the proctoring system aspect, it is noted that most of the existing research is commercially driven, with limited academic contributions. To optimize the proctoring system for online exams, we recognize the significant value of examinees' eye gaze and define important regions on and off the screen through calibration using "magic pixels". Moreover, we attribute cheating criteria using a formulated equation that takes into account factors such as Count, Frequency, Duration, and Regression. Two potential approaches for the proctoring system, namely Thresholding and Machine Learning (ML), are considered. However, our focus lies on the development of a thresholding-based approach. Overall, this thesis presents a comprehensive exploration of academic dishonesty in online examinations, proposes a proctoring system using eye gaze tracking technology, and compares the performance of different models and methodologies. The findings contribute to the advancement of proctoring systems and provide insights for the development of more effective measures against academic dishonesty.

Keywords - Smart Proctoring Systems, Academic Dishonesty, Malicious Intent Prediction, Neural Networks, CNN, Appearance Based Models, Eye Gaze Tracking

Acknowledgements

We would like to express my heartfelt thanks to Allah Subhanu Wata'ala for giving us the strength to finish this study and being with us when no one else was. We are also grateful to our loved ones for supporting us from the beginning to the end of our research. Lastly, we would also like to extend our sincere gratitude to our thesis supervisors Dr. Hasan Mahmud and Dr. Md. Kamrul Hasan for their constant encouragement and support throughout our study. We would not have been able to complete this research without the guidance and assistance of several individuals who contributed in various ways. We would like to acknowledge and appreciate their valued help.

Contents

List of Figures

List of Tables

Dedicated to Our parents for their continuous encouragement of my academic endeavors and research . . .

Chapter 1

Introduction

Academic dishonesty, such as cheating in online examinations, has always been a prevalent concern. There are no physical invigilators to monitor them. The only reliable source is to have the examinees send a constant video feed and have a one or more invigilators to monitor them through it. This approach obviously has many drawbacks. Firstly, it is very difficult to discern if an individual is attempting any malpractice from simple a video of their front face. Their gaze might be pointing at some other place even if their head position is stationary. Secondly, accurately predicting the eye-ball's gaze is very difficult for humans. This is more of an issue if the camera in use is of low resolution. Lastly, one invigilator may need to monitor numerous examinees' video feeds simultaneously. All of this makes ensuring academic integrity when conducting examinations online very difficult. Furthermore, there is also constraints from the examinees's sides as well. They may not have a camera of acceptable quality. Blurry videos can make monitoring them very difficult. In addition, some internet connectivity is also an issue. Some people may not have access to uninterrupted internet connection and thus if they get disconnected mid examination, there is no way to ensure that the candidate did not partake in any unethical means.

Despite these problems, many organizations used online examinations as a part of their evaluation process. However, after the Covid-19 pandemic, the world had to adopt this as an alternative to physical examinations. Now, even after the pandemic has ended, many organizations, even those that are outside of the education sector, are opting for such remote assessments due to their convenience and low cost.

1.1 Academic Dishonesty in Online Examinations

Academic institutes should be very concerned with academic dishonesty. It allows equity, character development, the task of imparting knowledge, student morale, faculty morale, future student behaviour, reputation for learning, and public trust in higher education [11].

If given the opportunity, examinees will partake in academic dishonesty to improve their grades. Because of this, when educational institutes decided to adopt to remote examinations, many students did partake in very unethical means [12, 13]. This is mainly due to the invigilator not being there physically. They may also take advantage of their hardware, like setting the camera at a certain angle or placing another device, like a smart phone, close at hand [11]. They may also have other people in the room to help them [11, 13].

1.2 Smart Proctoring

To prevent examinees from freely misusing the online platform to conduct dishonest means, a smart proctoring system is necessary. It should eliminate the need for an invigilator entirely by analysing all the different video feeds and flagging anyone it deems suspicious [14, 15, 16]. This should make it significantly more difficult for an examinee to partake in unfair and dishonest means. While there are many criteria for flagging an examinee as suspicious, our research focuses on leveraging eye gaze tracking technology to detect suspicious behavior during examinations. The reasons for employing eye gaze tracking methods will be discussed in subsection 1.2.1.

While the system described above should eliminate most problems related to the invigilator, other issues still do persist. A prominent one is that examinees will need a constant internet connection. One solution to this could be do install the proctoring system locally on each examinees' device and let it run on each of their devices during the examination. Although this does create a larger overhead for the participants, it will eliminate the need of a constant internet connection. The video feed will can stored and analysed locally. After the exam, it can be uploaded onto the online exam platform's server. If the video feeds are too expensive to upload, another solution can be designed where the locally installed version sends a report after analysing the video feed locally.

1.2.1 Eye Gaze Tracking

For our design, we have selected to study eye gaze tracking using appearance based deep learning models. The reasons for selecting these properties are discussed below.

1.2.1.1 Comparison with Binary Classification Methods

As mentioned above, our method of detecting unfair means is through tracking the examinees' eye gaze and recognizing cues that would indicate potential cheating or other deceptive practices. While most other works prefer binary classification [17, 18], these binary classification methods mainly involve analysing a frame from the video frame and by passing it through a trained Convolutional Neural Network which will identify cheating or other deceptive practices and flag accordingly. The main drawback here is that this is not customizable from the end users. After we develop the system, the people conducting the exams may wish to set their own criteria for flagging examinees. For instance, looking away from the screen for 1 second or making zig zag eye gaze patterns outside of the screen. The problem with using a model that uses binary classification is that changing the conditions for flagging someone is not that easy for these end users. However, if we design a system that has two modules: 1) the eye gaze tracking module concerned with tracking where the examinee is looking at and 2) the proctoring module concerned analysing the output from the gaze tracking module and determining if the data shows that the examinee is cheating. By separating their concerns into two differnt modules, we can design the system in a way for the end users to make modifications to the flagging requirements in the latter module.

1.2.1.2 Comparison with Geometry Based Methods

Geometry based solutions for eye tracking are very common [19, 20, 21]. They involve detecting the pupil of the subject and using mathematical calculations to project their gaze onto a point on the screen. The drawback is that it requires a good quality camera with very high resolution to detect the subjects' pupils. Recent works have also found that appearance based models when given enough training data can give accuracy comparable to geometry based solutions [22, 23]. Thus, if these models were trained with low quality video frames and given that the training data is large enough, it should produce comparable accuracy.

As for speed, geometry based solutions maybe computationally faster. However deep learning approaches and CNN-based solutions can be quite fast if the size of the input data is small [23]. This now gives us another criteria: the examinees' video feed should be of low resolution. If the size of each frame is small, CNN models can analyse it much faster. An in depth explaining of CNN models can be found in the appendix section B.4.

In addition, appearance based or end-to-end based solutions, if trained with the appropriate data, can account for variances like subjects wearing spectacles, varying pupil color or eye lenses, and differences in lighting [17, 22, 24]. For geometric models, these variations can greatly impact their performance. The only way to adapt to such changes is for the developers to design different methods for their pupil detection and vector projection depending on the circumstances. For instance, if the subject is wearing spectacles, the the mathematics for generating a gaze projection from the pupil will differ from that of someone without wearing spectacles.

1.3 Challenges

1.3.1 Gaze Tracking Challenges

There are a few challenges we need to consider when designing a solution incorporating gaze tracking. As mentioned in subsection 1.2.1.2, the system will find examinees wearing spectacles or have different colored pupils [25]. Some may even wear contact lenses. Moreover, the training of end-to-end models is very expensive, especially without high end hardware [22, 23]. In addition, our system will be used by students and most students will also not have access to high end hardware like a GPU [26]. Therefor, our gaze tracking solution needs to be optimized and have low computational requirements so that a user with an everyday laptop or computer can use it. The small size of the video resolution will help in this regard, however, that will require more time to train [22, 23, 25].

1.3.2 Proctoring Challenges

There are also a few issues when it comes to designing a smart proctoring system. Besides the hardware constraints mentioned above, the examinee may also not have access to a proper internet connection. When designing the system, we have to consider this possibility and ensure that the exam can still be conducted even if the examinee is disconnected from the internet. Besides this, there are other concerns for instance, dependency on camera's quality and position [27]. The position of the camera will also play a crucial factor as well [16, 25]. Lastly, there are also issues regarding privacy and breach of ethics that arises when we wish to monitor individuals through a camera [16, 27].

1.3.3 Research Challenges

Throughout this research thesis, we have also encountered several challenges that had a significant impact on the performance of our models and the outcomes of our experiments. One

such challenge was the limited availability and size of data sets specifically in the form of video data with low resolution. The scarcity of these data sets resulted in a shortage of dialogues for training and evaluation purposes. Another challenge was the computational cost associated with training and optimizing the gaze tracking models. The models that demanded extensive computational resources and proved to be computationally intensive. Furthermore, the absence of high-end hardware for model training significantly increased the costs associated with conducting the experiments. Lastly, our experiments required us to test these models on different hardware. Installing the model's dependencies on different computers and running them there was very time consuming.

1.4 Thesis Contributions

We proposed a system to incorporate gaze tracking into smart proctoring. We discussed how our designed smart proctoring system will use the gaze tracking module to identify suspicious behaviour. Also, we have conducted some experiments to find a suitable appearance based model for this specific system. Finally, we propose a final experiment to test our proposed system in actual examinations.

1.5 Thesis Outline

This Book is arranged in the given order: In Chapter 2, we have described the background studies. It contains discussions on the current status of smart proctoring and the status of online examinations and their proctoring methods. It also then talks about the work that has already been done on incorporating eye tracking and gaze estimation to invigilate examinees. In the chapter 3, we then have discussed our proposed methodology in terms of the architecture of the new system by describing the working mechanism of each module of the system and the procedure flow. In chapter 4, we discuss the how we designed our experiments. We go through the designs and objectives of both experiments. After that in the next chapter, we present our findings and Analysis the results. Finally, in chapter 6, we summarise our findings and research and and discuss some feature work that needs to be carried out.

Chapter 2

Literature Review

2.1 Smart Proctoring

Online exams have been increasing in popularity according to studies [14, 27]. This is mainly due to government imposing the necessary restrictions during the COVID-19 pandemic. The studies [11, 12, 13, 28] show that this has created an opportunity for students to engage in cheating. The study [14] argues that even after Covid, the popularity of online methods of conducting exams will still persist because of the many advantages it brings. Thus, there is now a need to create a system to safeguard against cheating for online exams.

There are many elements in smart proctoring. According to the study [27], they include

- Authentication
- Gaze Estimation
- Learning Management System Integration
- Foreign Object Detection
- Privacy and Ethics

Studies like [16] proposes the following techniques

- Bio metric solutions for authentications
- Object Detection / Audio Segmentation Algorithms to prevent other people from helping the examines.
- Browser extensions to detect if the examine is looking up anything online

- Tracking gaze to identify if the examine is looking at something else
- Recording the screen share

According to [14], some education providers were initially reluctant to adopt smart proctoring due to concerns with issues related to integrity and privacy. They also reported that there are technological barriers. Besides these issues, they found that some factors like the relative advantage and ease of use will greatly impact the likely hood of adopting smart proctoring. Other factors mentioned were compatibility with their existing systems to provide a easy process of integrating it into their current system. They also discuss the need for a trial and error based approach where they will need to be able observe the impact it will have and be able to make necessary adjustments to produce the required results.

The study showed that more than 55% of students in the study had a positive attitude towards their system, while 24% had a negative attitude due to network issues and 21% had a neutral attitude. They argue that this can be considered a viable alternative to traditional in-person exams, however more research is needed to fully understand its benefits and limitations. According to [14], Some areas that are still being researched are in the domain of AI-based video and audio analytics to detect cheating patterns. Other domains include image recognition to prevent impersonation, alerts based on suspicious behavior, student privacy concerns, etc. It also discusses the divide between those who favor authentic assessments and those in positivist or STEM disciplines who prefer exams.

The paper [29] addresses the increasing adoption of remote proctoring and examination software in education, particularly in response to the COVID-19 pandemic. It highlights three main concerns associated with these software packages: exam integrity, exam procedural fairness, and exam-taker security and privacy. To investigate these concerns, the authors conducted a systematic, technical analysis of proctoring suites used in U.S. law schools and state bar exams, focusing on the field of law to provide comprehensive insights. They identified four primary proctoring suites: Examplify, ILG Exam360, Exam4, and Electronic Blue Book.

Through reverse engineering, the authors discovered vulnerabilities in these proctoring suites, compromising their claimed security. They evaluated the suites from the perspectives of different adversaries, including law students, students with computer science experience, and experienced reverse engineers. The analysis revealed that the proctoring suites installed highly privileged system services with access to user activities, raising concerns about user privacy. Additionally, the facial recognition classifier used by Examplify to authenticate students showed accuracy concerns when tested against state-of-the-art classifiers. The authors also found variability in the performance of the classifiers across different racial groups.

The paper offers recommendations to improve the integrity and fairness of remotely proctored exams while mitigating privacy risks for students. It suggests privacy-protecting measures

for proctoring administrators and provides suggestions for vendors to enhance exam integrity without compromising student privacy.

2.2 The need for Proctoring in Online Exams

The paper [11] provides a comprehensive survey of cheating strategies, cheating methods, and cheating detection methods in online exams, with a specific focus on the Covid-19 pandemic. Covering a wide range of cheating strategies, the paper addresses issues such as lack of academic skills, lack of motivation, lack of self-regulation, lack of moral values, peer pressure, and perceived benefits and risks. Additionally, the paper describes common cheating methods, including impersonation, collusion, plagiarism, unauthorized materials, unauthorized devices, unauthorized software, and unauthorized websites. In terms of cheating detection, the paper reviews various methods such as biometric techniques (face recognition, iris recognition, fingerprint recognition, etc.), online proctoring (whether human or automated), lockdown browsers, challenge questions, and text originality checks. Overall, the paper serves as a valuable resource for individuals ranging from beginners to experts, providing a solid foundation for addressing assessment dishonesty in online exams and working towards its reduction.

Another case study [12] studied the concerns about cheating during the switch to virtual education due to the COVID-19 pandemic in Spain. To tackle this, the authors performed the following. They first implemented requests from students for longer time to take exams and asynchronous availability in a specific undergraduate course. They then analyzed records from the virtual learning environment to determine if cheating occurred. The analysis was done through a python tool they themselves developed. Besides that, they also incorporated other tools like Disco and Py-Cheat to detect evidence of cheating by analyzing the start and completion times, grades, and IP addresses of students who took the exam. Their results showed that some students cheated, and the authors suggested focusing on evidence-based assessment in the future. They concluded that evidence-based assessment is necessary to ensure the integrity of online exams and recommended improving the integration of the custom tool with databases and the virtual learning environment to facilitate this.

The study [13] examined whether cheating occurred in online economics courses by comparing the explanatory power of a prediction model for exam scores between proctored and unproctored exams. The authors conducted a study using data from two online courses in principles of economics to examine the relationship between exam scores and student characteristics. The results showed that the explanatory power of the prediction model was lower for the unproctored exams, suggesting that cheating was more likely to occur in the unproctored exams. The authors also analyzed the relationship between student characteristics and exam scores in the two courses and found that older students and those with higher GPAs had higher exam scores.

The authors used other methods, such as the Goldfeld-Quandt test and predicted scores based on the proctored exam model, to support their conclusion. These findings suggest that proctored exams may be necessary to ensure the integrity of online courses.

The paper [28] delves into the impact of social norms and peer effects on academic integrity during online exams in the context of the Covid-19 pandemic. Through a randomized field experiment involving nearly 500 undergraduate students from a major Spanish university, the paper examines the behavior of students taking an unproctored online multiple-choice exam without the ability to backtrack. To study cheating behavior, the paper utilizes the variations in the order of exam problems and collects detailed data with timestamps. The findings of the study reveal that in the later round of the exam, the number of correct answers to questions was 7.7% higher compared to the earlier round. Additionally, the average completion time for questions in the later round was 18.1% shorter than in the earlier round. Interestingly, the paper also investigates the impact of a reminder of the university's code of ethics, which was sent to a subgroup of students midway through the exam. However, the study concludes that this reminder did not have any significant effect on the levels of cheating observed.

2.3 Proctoring With Gaze Tracking

There have already studies in implementing gaze tracking to achieve a number of specific tasks. In the study [30], the authors used a convolutional neural network (CNN) to estimate gaze coordinates using images taken with an unmodified web camera. The data collection method involved using a desktop application to capture images of test subjects using a built-in laptop camera while they clicked on different locations on a screen. The subjects sat in front of a computer screen and clicked on 20 points displayed on the screen in various locations. After each click, a series of three camera images was taken and stored with information about the point's location. The images were then pre-processed and fed into a CNN to predict where on the screen the subject was looking. The authors used image processing techniques to preprocess the data and a Viola-Jones classifier to detect faces in the images. They trained and tested several different CNN architectures on the data set and found that it was possible to predict human gaze points with reasonable accuracy using an unmodified camera. The results of the study [30] showed the following.

1. It is possible to predict human gaze points from an unmodified camera with reasonable accuracy.
2. The best results were achieved when the network was trained on sharpened images and when the entire face was used as input.

3. The results were not as accurate when using only one eye as input.

4. It was found that using a CNN with a carefully designed topology and hyperparameters can allow for reasonable results in a real-world environment.

The authors also discussed the potential for future work in this area, including improving the robustness of the method to lighting conditions and expanding the data set to include a wider range of gaze locations. However, the study had some limitations, such as classifying gaze points into 20 areas rather than finding the exact gaze point and a limited number of participants and data. In future research, the authors plan to address these limitations by expanding the data set and testing the model on publicly available data sets.

Another paper [18] presents an automated online proctoring system using attentive-net to assess student mischievous behavior during online examinations. The system uses four components to detect malpractices: face detection, multiple person detection, face spoofing, and head pose estimation. The system uses a combination of attentive-net, face net, liveness net, and solvePnp equation to perform these tasks. The system is evaluated on CIPL data sets and customized data sets with various types of malpractices. Their system achieves an improved accuracy of 0.87 and demonstrates that it is more accurate, reliable and robust for proctoring system that can be practically implemented in real time environment as automated proctoring system.

A frame work was also designed in [6] for automated monitoring of candidates during examinations. The system used a webcam to analyse the video frames of the students. The analysis is done on the students own system, eliminating the need for constant high bandwidth internet connection. Their designed version of an online proctoring system is showed in Figure 2.1.

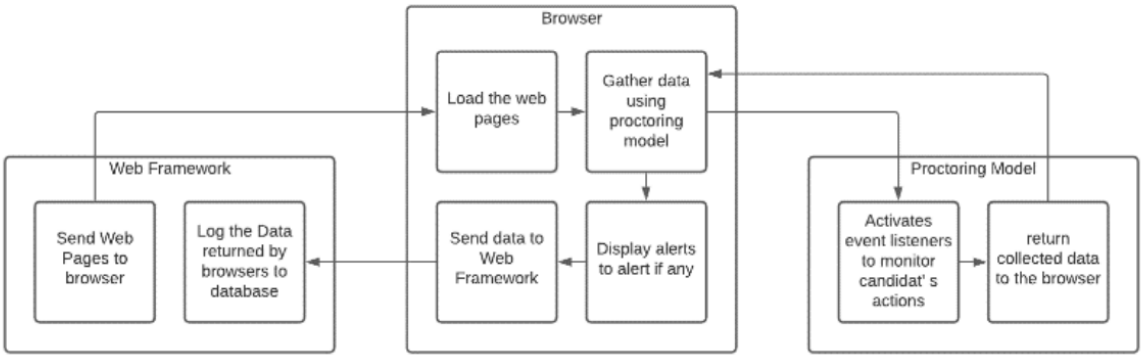

FIGURE 2.1: The architecture designed in [6]

Some studies have incorporate calibration into their design as well. In the proposed methodology of [31], calibration plays a crucial role in utilizing eye-tracking technology for online exam cheating detection. Their system provides online proctoring though the following steps.

1. Eye Gaze Model Calibration: Users calibrate the eye tracker by clicking on red dots, which helps tune the gaze estimation model.
2. Screen Corner Detection: Gaze readings are taken to determine the screen corners, enabling detection of suspicious gazes outside the screen.
3. Continuous Proctoring: Eye-tracking data is continuously collected during the exam to identify cheating behaviors. A warning is issued if cheating is detected, and a full report is generated for the examiner.
4. Continuous Proctoring: Eye-tracking data is continuously collected during the exam to identify cheating behaviors. A warning is issued if cheating is detected, and a full report is generated for the examiner.
5. Cheating Detection Model: An unsupervised cheating detection engine using OCSVM is employed to identify outliers in gaze patterns, indicating potential cheating instances.

Overall, calibration ensures accurate gaze estimation and enables the detection of abnormal behaviors during online exams.

Lastly, the work in [15] proposed three techniques of detecting unfair means. The first technique focuses on tracking the direction of the students' heads, specifically when they deviate from their initial direction towards the exam script. The second technique involves monitoring the movement of the students' iris. Lastly, the third technique is utilized to identify contact between a student's hands and face, as well as between different students, to detect shared abnormal behavior.

2.4 Gaze Estimation

The authors of [14, 16, 27] believe that gaze tracking can be an integral element for any online smart proctoring system. There have been many works done in this domain. The study [32] has identified three main ways to perform this task.

1. **Geometry-based solutions:** use information on eye shape, including eye open-close states, eye size, and eye color, to determine gaze points. Requires multiple cameras (usually) and changes in illumination decreases accuracy.
2. **Feature-based solutions:** attempts to extract distinctive features such as contours, eye corners, and corneal reflections or glints. Such methods work well in controlled indoor environments. However, small eyes or facial attachments such as glasses, hats, and eyelashes can degrade gaze tracking performance.
3. **Appearance-based solutions:** Inspired by recent developments in object detection and recognition with deep learning. Recent Appearance-based techniques include convolutional neural networks (CNNs) for gaze estimation. The main advantage is that it can be used in everyday settings and an uncontrolled environment.

Additionally The paper [23] introduces a novel hybrid neural network model for automatic detection of depression using 128-channel resting EEG signals. This model combines a 2D Convolutional Neural Network (2DCNN) and a Long Short-Term Memory Network (LSTM), enabling it to capture both spatial and temporal features of EEG signals. To evaluate the model's performance, a data set of 24 participants with depression and 24 healthy controls is utilized, employing a 24-fold leave-one-out cross-validation approach. The model achieves an impressive average classification accuracy of 95.1% and an AUC of 0.98 for detecting depression in 6-second participant EEG signals. Notably, it also exhibits a 100% probability of correctly classifying the EEG signals of 300-second participants. Comparatively, the proposed model outperforms other baseline methods, including Support Vector Machine, K-Nearest Neighbor, and Decision Tree.

For geometry-based solutions, the research in [19] aims to develop a low-cost, convenient eye gaze tracking system using a web camera in a desktop environment. The system tracks the human face in real-time video to extract the eye region and uses virtual reference points and head movement information to estimate eye gaze. The system was trained using a least-squares method to transform pupil position to screen coordinates and can estimate gaze location in real-time. The system uses Haar Cascade face detection and eye detection and is able to remove spurious detections using known face geometry to produce reliable bounding boxes around the user's eye. It also uses a gradient-based method to find the pupil center within the bounding

box for each eye and calculate the gaze direction. The system was tested and found to have an average error of approximately 5-25 cm in favorable conditions.

Another much older work can be found in [21], where they propose a system that utilizes the stationary position of the webcam relative to the head to accurately track eye movement. The proposed approach starts by applying a dynamic threshold to binarize the image, then extracts geometric features of the eye from the binary image. Using an estimation method based on the geometric structure of the eye, the positions of the two eye corners are detected. Subsequently, the center of the iris is located by comparing image contours with an iris boundary model. Finally, using the relative position of the iris center and eye corners, the position of where the eye is looking on the monitor is calculated. This system is cost-effective as it only requires a low-cost webcam and a personal computer. The experimental results demonstrate the ability of the proposed system to accurately detect eye movements in real-time.

Feature-based solutions include [20] which proposes a system that can be used with a general low-resolution webcam. The paper explains that traditional research methods for eye tracking often use expensive equipment or infrared-based techniques. However, with the increasing need to analyze user behavior by tracking eye attention in general applications, it is no longer practical to use these traditional methods. The paper describes the techniques used in the proposed system, including an illuminance filtering approach to remove the influence of light changes, a hybrid model combining the position criterion and an angle-based eye detection strategy to locate the eyes accurately and efficiently, and the use of the Fourier Descriptor to describe the appearance-based features of eyes compactly and the Support Vector Machine to determine the eye-gaze position. The paper claims that the proposed algorithms have high performances with low computational complexity

Another feature-based approach is used in [33]. The paper presents a method for real-time eye gaze tracking using a low-cost webcam in a desktop environment. Their proposed system tracks the human face in real-time video sequences to extract the eye regions, combines intensity energy and edge strength to locate the iris center, and uses a piecewise eye corner detector to detect the eye corner. A sinusoidal head model is used to simulate the 3-D head shape, and an adaptive weighted facial features algorithm is used to estimate the head pose. The eye gaze is then tracked by integrating the eye vector and head movement information.

Finally, as for appearance-based solutions, there has been a significant amount of research in recent years, particularly with the advent of deep learning and neural networks. One notable example is the work by [1], which trained a Convolutional Neural Network (CNN) using low-resolution (VGA quality) images for mobile applications. To train the model, the team created their own data-set through crowdsourcing. Other studies, such as [32] and [34], have built upon this work using techniques such as Long Short Term Memory (LSTM) networks to improve computational efficiency and add other features like estimate the distance from the screen.

Overall, these studies demonstrate the potential for using deep learning and neural networks to improve the accuracy and efficiency of appearance-based solutions for eye-gaze tracking.

The research in [35] focuses on improving the accuracy of gaze point estimation by using videos as input data, rather than just images, and incorporating both spatial and temporal features. A convolutional neural network (CNN) is used to extract spatial features, while a long short-term memory (LSTM) network is used to capture temporal features. The resulting CNN Concatenating LSTM network (CCLN) is tested and optimized using various techniques. The study also proposes a method for constructing data sets of videos for gaze point estimation and compares the performance of the CCLN model with existing CNN-based methods using these data sets. The results show that the proposed CCLN model performs better than other methods, with an accuracy of 93.1% for the best model and 92.6% for the general MobileNet model.

The study [36] investigates the use of unmodified web cameras to track a person's gaze for low-cost eye tracking. The authors trained and tested a convolutional neural network (CNN) with various architectures using images from web cameras. They found it possible to predict human gaze points from an unmodified camera with reasonable accuracy. They created and tested a data set collected in natural, uncontrolled environments using standard laptops and webcams. Future research aims to address the limitations of the current study, including utilizing publicly available data sets and expanding the data set.

Chapter 3

Proposed Methodology

Our methodology involves studying existing literature and designing a proctoring system that satisfies our requirements. By accurately tracking the examinee's gaze, the system should possess the capability to predict instances of cheating. Our overall design is discussed in section 3.1 where the we discuss how our two primary components: the gaze tracking module and the proctoring module interact together to flag suspicious behaviour. Our system relies on a calibration system, which is also discussed in depth in section 3.2 The criteria for detecting such unfair means is also discussed in section 3.3. Furthermore, in addition to accurately tracking the examinee's gaze, it is essential for the system to provide fast predictions with minimal latency. This will ensure that potential instances of cheating can be swiftly identified and flagged. Besides these two main requirements, our system should be usable by the every day general student. Therefore, the algorithms used to track the gaze and eyes should be optimized for any everyday computer's processor. This is also discussed further in section 3.4, Finally, we also discuss in-depth how the gaze tracking module will work and what are its requirements in section 3.5.

3.1 Overall Architecture

Our proposed research aims to establish an environment where the examinee can use their web cam to send a video feed to our system which will then predict if the person in question is attempting to cheat. The setup involves strategically positioning a webcam in front of the subject to ensure continuous capture of their entire face throughout the examination period. To facilitate real-time unfair means detection, the software application will need to be installed locally onto the examinee's system as discussed in section 1.2. This will enable the analysis of data in real-time within the examinee's own environment without the need of a constant internet connection or external invigilation.

The architecture of our system can be divided into two parts

1. The Gaze Tracking Module
2. The Proctoring System

The design of the gaze tracking module is illustrated in figure 3.1. The web cam is placed directly above the screen and sends real-time video feed to the gaze tracking module. The module does some pre processing for each individual frames received from the web cam and puts it into the gaze tracking model. The pre-processing or preparation of each frame can differ due to the gaze tracking model or algorithms used. The output from these models will always be the x and y coordinates of the examinee's gaze respective to the camera's position.

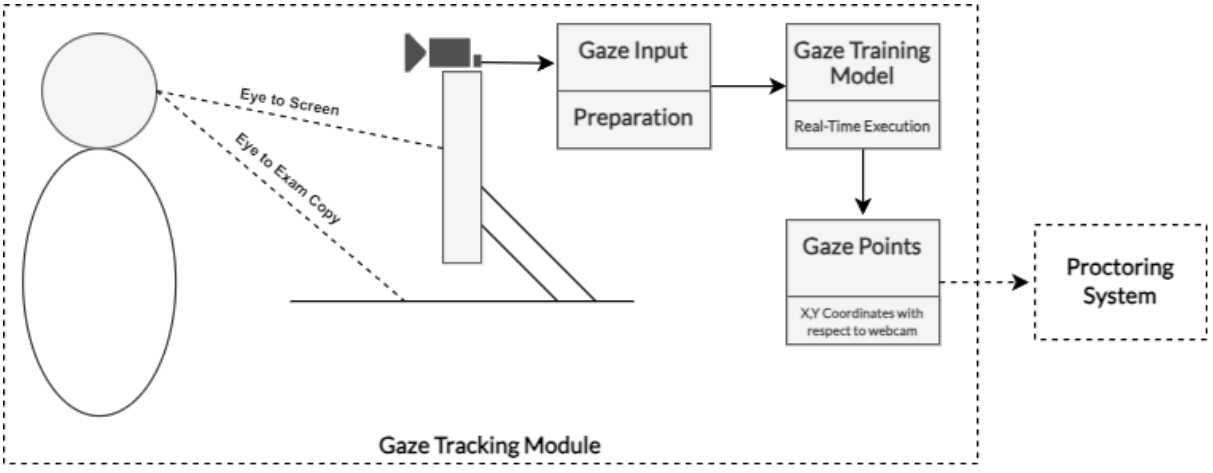

FIGURE 3.1: The Gaze Tracking Module

Next, figure 3.2 shows how the proctoring system works. It will take the gaze points from the gaze tracking module and predict whether the examinee is cheating. It will need to make these predictions in real-time as well. To make these predictions it will analyse the subject's gaze points and identify cues of unfair means. To identify the cues, the system needs to be calibrated so that it knows the thresholds related to the verices of the screen and script. From there, it generates a restricted zone which is the region where the examinee's gaze should not be in. When the gaze does fall there, it will check if it satisfies any of the criteria for unfair means which are, as shown in the figure, *time, frequency* and *regression pattern.*

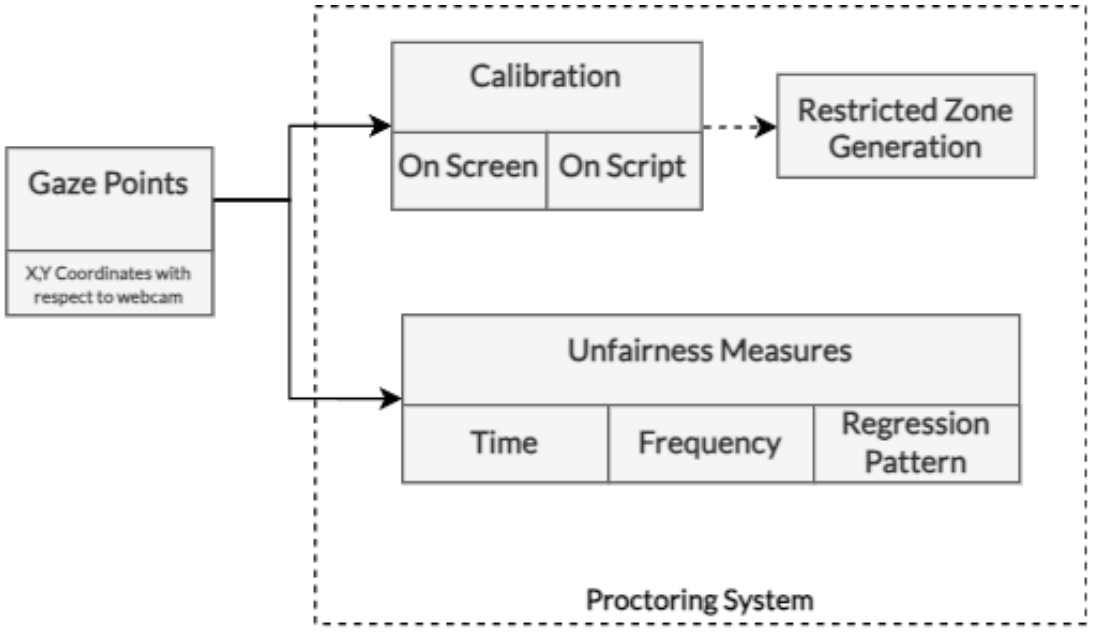

FIGURE 3.2: The Proctoring System

Following the conclusion of the exam, a comprehensive report can be generated and shared with the examination organizers, providing detailed insights into the level of suspicion detected in each subject. This report encompasses a comprehensive analysis of the collected data, highlighting any abnormal behaviors or potential instances of cheating identified during the examination. By providing such a detailed report, the system offers valuable information to the organizers, enabling them to make informed decisions regarding further investigation or actions as necessary. The generated report serves as an essential tool for maintaining the integrity and fairness of the examination process, ensuring that appropriate measures can be taken based on the identified suspicious activities.

3.2 Calibration

As discussed in subsection 1.2.1.1, our study will only involve detecting unfair means through tracking the subject's gaze. There have already been many works featuring the characteristics of subjects attempting to cheat. One of the most reliable method is to track if they are looking beyond a certain threshold [31]. Our system could track their gaze and flag them as suspicious if their gaze falls out of a certain bounded range. However setting a general range can be quite difficult due variations in the device and environment of the test takers. For examples, some examinees will have a smaller screen while some may seat further away from their table. To account for such variations, we propose a simple calibrating method. We will first ask the participant to calibrate the proctoring system and the system will ask them to look at specific points, first on the four vertices of their screen, and then on the four vertices of their exam scripts. Figure 3.3 shows the four corners the user will look at and how the system will generate a bounded restricted zone around it. The user will be asked to to look at each of the calibration points and press a button to prompt the system to generate a marker at that point. The four vertices identified can be considered as *"magic pixels"* as they are the points we are most concerned about. The region beyond these points is where we will want to identify characteristics of unfair means.

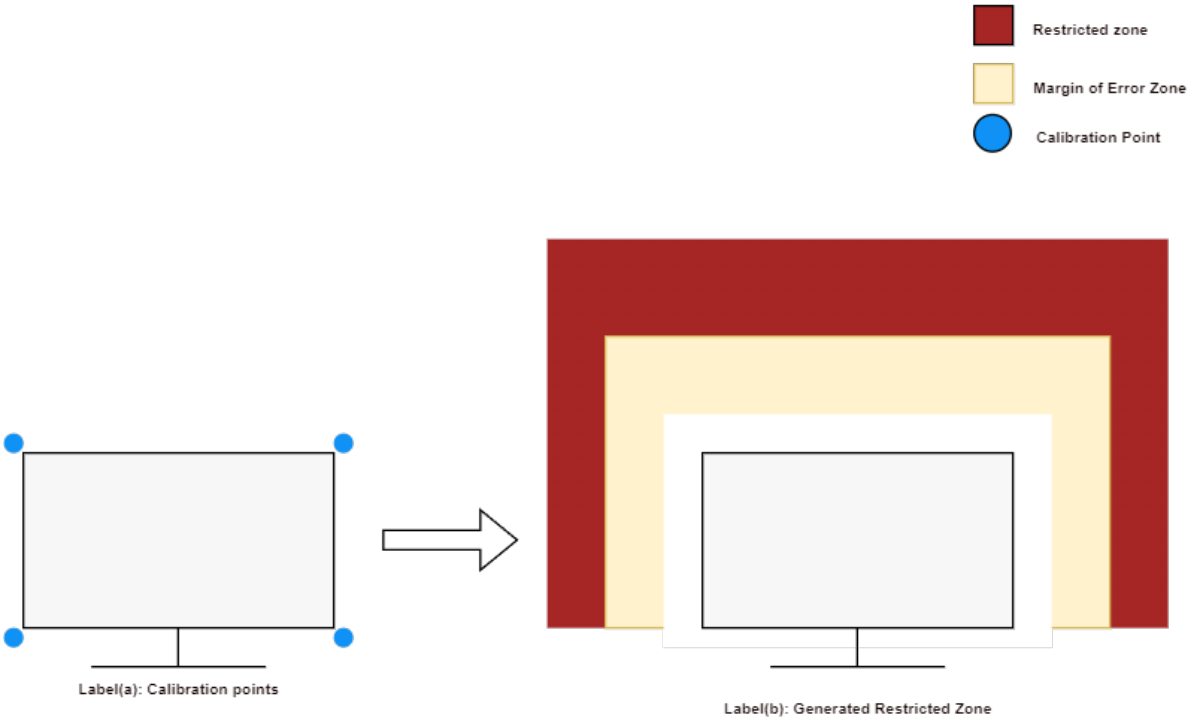

FIGURE 3.3: Generating restricted region around the screen

While there are some works that used calibration to find *"magic pixels"* at the vertices of the screen to generate a resticted zone [31], there is not that many that address hand-written script based examinations. For such scenarios, we propose that we do the same for the script as well, i.e., calibrate it using the proctoring system to find the vertices of the script. This is illustrated in figure 3.4. When all four vertices of the screen are recorded, the system will then ask the examinee to look at the four vertices of their exam scripts exam scripts in a similar way. It will then record the four vertices of the exam script and generate the required bounded regions as shown in the figure.

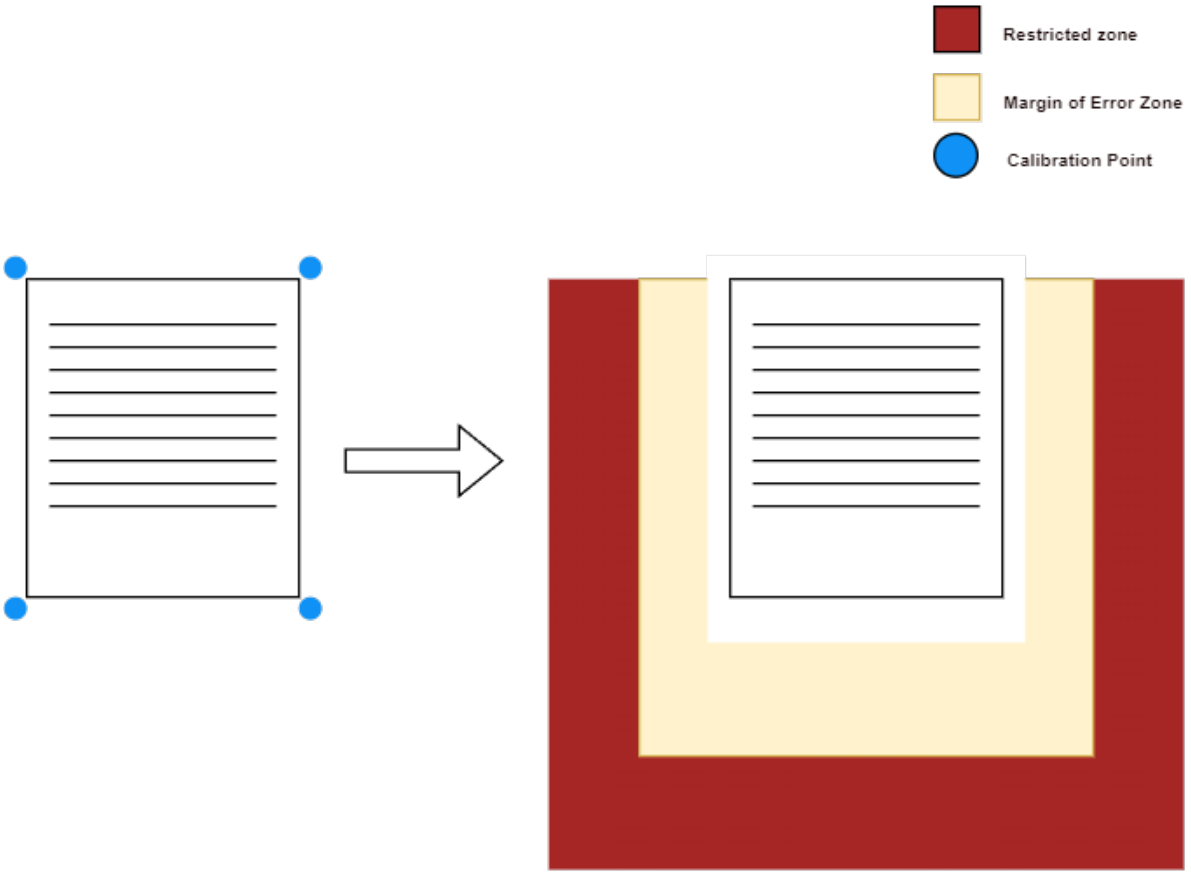

FIGURE 3.4: Generating restricted region around the script

It should be noted that for both process of calibrating for the sceen and script, there is a region marked as *"Margin of Error"*. This region allow the gaze tracking model to make errors and its size will directly correspond to the used gaze tracking model's accuracy.

3.3 Unfair Means Detection

Now that we have defined the restricted region, the subsequent step is to accurately distinguish between genuine instances of examinees looking away from their screens or scripts and situations where cheating might be occurring. It is imperative to defferentiate between the two as an innocent gaze movement outside the screen can occur and create false positives. Below, we discuss some cues that can be used to determine if the subject in question is actually cheating when their gaze falls in the restricted region.

3.3.1 Time Restrictions

In order to avoid falsely flagging examinees as suspicious for momentary gaze in the restricted region, it imperative to implement a time restriction mechanism. This time restriction defines an acceptable duration for the user's gaze to remain within the restricted region without triggering suspicion. A paper designed their own proctoring system suggests that the time period should be longer at around 1 to 3 seconds [15]. Another work suggests that a time duration of 1 second should be considered as noise [31].

By incorporating this time restriction, the proposed system ensures that only prolonged or sustained periods of gaze within the restricted region are considered as potential indicators of suspicious behavior. This approach aims to strike a balance between accurately identifying suspicious activities and minimizing false positives, thereby enhancing the reliability and effectiveness of the cheating detection system.

3.3.2 Frequency

This is the number of times the examinee's gaze was on the restricted region during a period of time. This period of time can be set to around 1-2 minutes.

3.3.3 Regression Patterns

It is also important to note that reading causes the eyes to move in a certain pattern similar to regressions [37]. This regression pattern is explained more so in-dept in the appendix section A.1. Therefor, our system could also be implemented to flag such patterns if they do occur in the restricted zone as that would clearly mean that the examinee is reading some sort of textual information placed there.

Hence, to enhance the effectiveness of our proctoring system, it is essential to implement a mechanism that flags examinees when their gaze repeatedly performs regressions or exhibits similar movements within the restricted zone.

3.3.4 Gazes outside the Natural Viewing Range

The typical human has a viewing angle of around 120° horizontally and 130° vertically [38]. A typical human can see between these ranges without moving their head pose. A more detailed expansion of human virwing angle can be found in appendix section A.2. Thus, if any examinee tries to look beyond these ranges, it should indicate that there is a specific object of interest that requires attention. This, however, does not necessarily indicate an attempt at academic dishonesty. Some works suggest that issuing a simple warning at first should suffice [6, 30]. Repeated attempts should, however, be flagged as suspicious.

3.3.5 Overall Criteria

Using the above thresholding criteria, we can use generate an equation to predict if the examinee is cheating. We will denote this prediction as P. During the whole exam session, the system will keep track of the examinee's gaze and look for certain criteria.

$$P = W_1.t + W_2.f + W_3.N_R + W_4.N_H$$

The first cirteria is the total time the gaze was in the restricted zone in a predefined time period. This variable can be denoted by t. Do note that momentary gazes of less than 1 second on the restricted zone should not contribute to this time [31]. Next is the amount of times the gaze was in the restricted zone within the time period. This is denoted by f. Momentary gazes of less than 1 second should be ignored here as well. We wiil also look for regression patterns in the gaze when it is in the restricted zone. The number of times regression movements were detected can be denoted with N_R. Finally, we will also count the number of head movements during this period. This can be denoted by N_H. All these variable are used to construct the equation above. The values of the weights W_1, W_2, W_3 W_4 will need to be determined through a series of experiments that will be discussed in the following chapters.

3.4 Need for Optimisation

Our proctoring system should be designed to run on any system, even those with low-end hardware. Most students do not have high quality cameras. A survey also revealed that students at a particular university rarely have GPUs or other high-end hardware [26]. Therefore, it is imperative that our system should be able to track the subject's gaze with reasonable accuracy and speed even while keeping the hardware requirements low. This will require very computationally optimized algorithms.

3.5 The Gaze Tracking Module

The gaze tracking module is an appearance based end-to-end model neural network. An explanation of an end-to-end neural network can be found in the appendix section B.3. It will be given a frame from the examinee as an input and will output the coordinates or vector location of the examinee's gaze in that respected frame. Our proctoring system will take these frames as inputs and determine if the examinee is attempting to partake in dishonest means.

The module will run locally along with the main proctoring system. This means that any everyday laptop or desktop should be able to run it and have an acceptable performance. Our criteria for performance are accuracy and speed. Accuracy is the measure of how close the predicted gaze is to the true gaze and speed is the measure of how fast the module is analysing each frame. It should be noted that very accurate deep learning models have more layers and neurons, and so, usually take longer to process images. Similarly, if speed is required, less layers and neurons would work, however accuracy would be traded off. To find the right balance between these two and make sure it runs on low-end hardware, we conducted a series of experiments with different appearance based gaze tracking models. We will discuss about these experiments in the next sections.

Chapter 4

Experimental Design

Our research requires a few experiments. The first is to find the appropriate gaze tracking module. Next, we have to create a prototype of our proposed design and train it using real time data from actual exams. Lastly, we will then have to conduct a usability study on our implemented system.

For the first portion of our experiment, where we had to searched for the most appropriate gaze tracking model. we established the following requirements: that it has to be accurate and fast enough to process the frames in real-time. However, we also need to keep it affordable for those who do not have high-end hardware. It is quite unreasonable to assume that most students have high end hardware like GPUs or high resolution web-cams. Our experiments involved selecting a set of publicly available end-to-end gaze tracking models and measuring their accuracy and speed. Accuracy is a measure of how close the predicted gaze point is compared to the actual gaze point. Speed on the other hand is the measure of how fast the the module is analysing each frame. Furthermore, we also had to verify if the speed is relatively acceptable when run on different systems with different hardware specifications to make sure that students with low-end computers would be able to run it locally. Section 4.1.1 describes how we setup our experiments and Section 4.1.2 discusses the the gaze tracking models that were used for our said experiment. Section 4.1.3 contains a brief explanation of the matrices we defined for our experiment. Section 4.1.4 discusses the data sets and section 4.1.5 briefly summaries how we trained and tested our model on the different systems.

The second portion of our experiment involves a prototype of our system. We will conduct a controlled experiment with simulating actual online examinations and have train our proctoring system to predict if the examinee is attempting to cheat using the features discussed in the previous chapter. We will set the values of the weights for each unfair means feature through logistic regression. Details can be found in section 4.2. Afterwards, we will take our system and use the weights we found and conduct a usability study. Details for this are in subsection **??**.

4.1 Experiment 1: Gaze Tracking Module

This is the experiment we conducted to find a suitable gaze tracking model that satisfied our requirements. The following subsections discuss how we planned and designed this portion of our work.

4.1.1 Methodology

To set up the environment for evaluating the accuracy and speed of our trained gaze estimation models on different computers, we followed the following these steps:

1. **Hardware Selection**: We selected a set of diverse computers with varying specifications that could to represent a range of hardware configurations. Considering factors were processor speed, RAM capacity, and graphics capabilities. For our experiment, we selected the following systems.
   (a) R5 2600 + RTX 2060, 16 GB
   (b) i5-12400 + RTX 3060, 32 GB
   (c) R5 2600, 16 GB
   (d) i5-6600km 32 GB
   (e) i3-3300M, 8 GB
2. **Model Selection**: We selected two publicly available models being iTracker [1] and L2cs-net: [2]. We will go more in detail about these two gaze tracking models in section 4.1.2. These two models were tested on the above systems and then their performances interms of accuracy and speed were compared.
3. **Defining the matrices to measure**: We defined which matrices to measure for each model. The method for selecting these have already been discussed in-depth.
   - Accuracy
   - Speed
   - Optimization
4. **Data Set Selection**: We had to select a common data set on which all of our models were to be experimented on. Section 4.1.4 will go more into detail about the publicly available data sets and which one was used.
5. **Model Deployment**: Transferring the trained proctoring models to each computer. We had copied the model to each computer, ensuring that the models are compatible with the deep learning framework and version installed on each computer.

6. **Analysis**: After we recorded our observations, we analysed our data to draw conlusions. Our findings can be found in the following chapter 5.

4.1.2 The Gaze Tracking Models

As mentioned, we used two models in our experiment: iTracker [1] and L2cs-net [2]. The following is a brief discussion of the architecture of the these two models.

4.1.2.1 iTracker [1]

The **Itracker** model was designed for mobile phones and thus is intended to run on very low hardware requirements. It uses extracts the regions of interest from the frame, namely eyes and face, and passes it through its variation of the Convolutional Neural Network (CNN). The end-to-end neural network then output the the gaze points for each frame. Figure 4.1 illustrates how the model works.

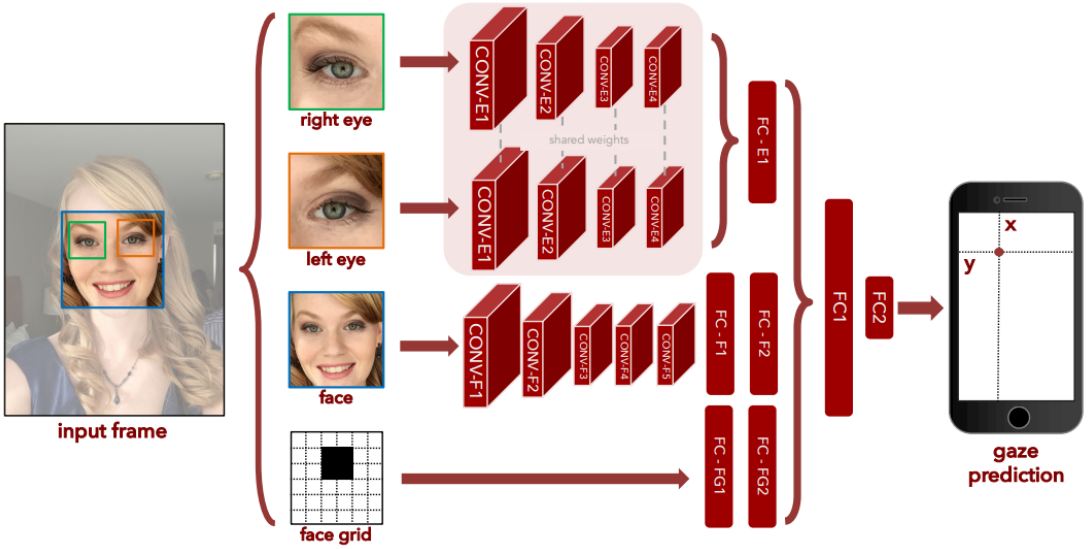

FIGURE 4.1: Overview of the iTracker Gaze Tracking Model [1]

The model uses a two-stream achitecture, where appearance and shape information are processed seperately and then concatenate and put into the model as input. This allows the model to utalize both appearance details aswell as the global facial structure, leading to improved accuracy.

4.1.2.2 L2CS-Net [2]

The **L2CS-Net** model uses the ResNet50 model as its base. Details about the ResNet50 model can be found in the appendix section B.7. It uses this model to generate the spatial gaze features from the frames and uses these to predict the horizontal and vertical angles separately. It also uses a probability algorithm to calcualate the likely hood of the values it predicts and, after getting the actual values, uses the mean square error for the backward propagation. This architecture is illustrated in Figure 4.2.

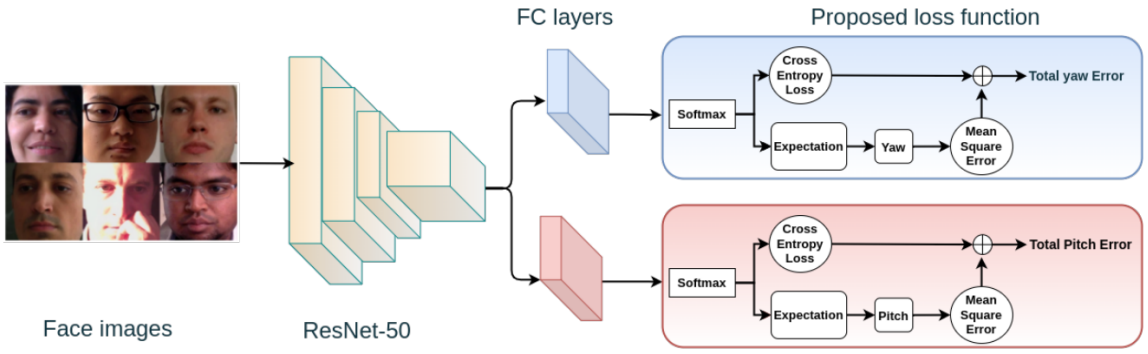

FIGURE 4.2: Overview of the L2cs-net Gaze Tracking Model [2]

It should be noted that this model has two separate loss functions for the vertical and horizontal error calculation. Each loss function starts with a softmax layer to convert the network output logits into a probability distribution. Afterwards, a cross-entropy loss is used to calculate the binary classification loss between the output probabilites and the target binary labels. Next, we calculate the expectation of the probability distribution and use that with the actual values found to calculate the mean square error. We add this mean square error from the predictions into the classification loss.

4.1.3 Evaluation Metrics

4.1.3.1 Accuracy

Accuracy score in machine learning is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made [39]. We calculate it using the equation below.

$$\textbf{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Number of Predictions}}$$

4.1.3.2 Speed

This is another evaluation measure of the rate at which the model makes predictions per unit of time. For our experiment, we will be using a series of frames as input and the output will be the prediction it makes. We will measure this with the number of processed frames per second or **FPS**.

$$\textbf{FPS} = \frac{\text{Number of Processed Frames}}{\text{Time Taken}}$$

4.1.3.3 Optimization

This is the measure of how well the model performs given the resources it has and is mainly used to describe the extent to which the performance will be constant when hardware resources are low. Our experiment requires us to make sure the gaze tracking model works on all system and especially on ones with low hardware requirements as discussed in 3.4. We measured it by running the tests on a set of different systems with varying hardware specifications and analysing the extent to which the performance matrices, i.e. *Accuracy* and *Speed* decreased.

4.1.4 Data Sets

The data sets used in our research include the Idiap data set, MPIIFaceGaze data set, and GazeCapture data set. These data sets provide valuable recordings of eye movements and gaze patterns from multiple participants, captured under various conditions. They have been widely used for training and evaluating gaze tracking algorithms and have contributed to advancements in the field of gaze estimation.

1. **GazeCapture** [1]: The GazeCapture data set is a large-scale data set collected by researchers at MIT. It consists of recordings from over 1,100 participants, captured using mobile eye-tracking devices. The data set includes a wide range of head poses, different lighting conditions, and various indoor and outdoor environments. The GazeCapture data set has been widely used to train deep learning models for gaze estimation and has contributed to advancements in the field.

2. IDIAP [40]: The IDIAP Data set is a widely used data set for gaze estimation research. It consists of recordings of eye movements from multiple participants captured using eye-tracking devices. The data set includes various conditions such as different head poses, illumination variations, and occlusions, making it valuable for training and evaluating gaze tracking algorithms under different scenarios.

3. MPIIFaceGaze [41]: The MPIIFaceGaze data set is another popular data set used for gaze estimation research. It was collected at the Max Planck Institute for Informatics and contains eye images and head pose of a wide variety of participants from a front camera. The data set includes recordings of 15 participants performing daily activities, providing a diverse range of real-world gaze patterns in a very controlled environment. The MPIIFaceGaze Data set has been widely used for training and evaluating gaze estimation models.

MPIIFaceGaze data set which is based on the MPIIGaze data set, with the additional human facial landmark annotation and the face regions made available. Facial landmarks annotations were conducted in a semi-automatic manner as running facial landmark detection method first and then checking by two human annotators. The pupil centers were annotated by two human annotators from scratch. The reasons for selecting this data set is discussed below.

- **Diversity and Real-world Scenarios:** The MPIIFaceGaze data set is known for its diverse set of real-world scenarios. It includes recordings of 15 participants engaging in various daily activities, providing a wide range of gaze patterns and environmental conditions. If your research aims to develop gaze estimation models that perform well in real-world scenarios, the diversity and realism of the MPIIFaceGaze data set may be a significant advantage.

- **Head-mounted Camera Perspective:** Unlike the Idiap and GazeCapture data sets, the MPIIGaze data set captures eye images from a head-mounted camera perspective. This perspective can provide a more natural and realistic representation of gaze behavior in real-world settings. If your research focuses on gaze estimation from a head-mounted camera perspective, the MPIIFaceGaze data set would be more suitable.

- **Variances in Participants** The data set contains a very wide range of participants for example, people with spectacles or contact lenses. There are even a considerable amount of frames with the participants looking away from the screen.
- **Data Set size:** It consists of recordings from 15 participants, and each participant's data contains approximately 37,667 face images.

4.1.5 Model Training

In order to train the two models iTracker and L2cs-net, we pulled the codes available. A study re implemented the iTracker model to be used on the MPIIFaceGaze data set [42] and we used their version of the code for our experiments. We had to use this code to train the model and using that model, we tested it against different systems.

For the case of the L2cs-net had a repository with a pre-trained model available and the required code already implemented to work on the MPIIFaceGaze data set. We simply had to follow the instructions on how to setup the pre-trained model and test it out. Similar to the previous model, this had to be done on the systems mentioned.

4.2 Experiment 2: Determining the weights of the Unfair Means Features

We have previously established that our proctoring system can predict if the subject is cheating using the following equation.

$$P = W_1.t + W_2.f + W_3.N_R + W_4.N_H$$

Here, P is the prediction of the subject attempting to cheat in that frame. t denotes the total duration the subject was looking outside the bordered regions in a period of 1 minute. f denotes the number of times the subject has looked beyond the bordered regions within a period of 1 minute. Lastly, N_R is the number of regression movements detected and N_H is the number of head movements for looking very far away from the screen. both in the period of 1 minute. The following experiment is designed to determine appropriate values for the weights of each of the above features, i.e., the values for W_1, W_2, W_3 and W_4.

This can be done through either linear regression or logistic regression. More information about both of them can be found in appendix section B.1 and B.2. We are proposing logistic regression as the value of P is categiral, i.e., has 2 discrete values of cheating or not cheating. We would re-model our initial equation for logistic regression as shown.

$$P = \frac{1}{1 + e^{-z}}$$
$$\text{where } z = W_1.t + W_2.f + W_3.N_R + W_4.N_H$$

To perform this, we will need a considerable amount of data. Due to the unavailability of such datasets, we will have to gather the data on ourselves. This involves gathering atleast 20 participants and have then take part in an online examination. We will ask a few of them to cheat and label their data likewise. This controlled experiment will produce the required data and our proctoring system will look for the unfairness features and use them to predict if the participant is cheating. Our logisctic regression function should help determine the appropriate weights for each feature. With this done, we will then conduct a usability study for our implemented system. The usability study will involve another round of participants different from before and we will ask them to give an online exam. We will also ask a few to cheat as well.

The final task is to evaluate if our proposed proctoring system is suitable for online examinations. To do so, we would have to conduct a usability survey with another different group of users.

Chapter 5

Results Analysis & Discussions

In the previous chapter, we designed two experiments. First, an experiment to find a suitable appearance based gaze tracking model for our proctoring system. We identified the parameters we will use to evaluate the models and how we used them to design our experiments.

In this chapter, we present an in-depth analysis of our experimental findings and a discussion on which model would be suitable for gaze tracking in our proctoring system. Section 5.1 show our actual results for each of our models in a tabular format and discusses said implications.

Our experiments saw that while it simple to make conclusion on which models gives more accuracy or which one performs faster, the degree of optimization is much more difficult to measure. We proposed to run the models on different systems with varying hardware capabilities and found that there are a few anomalies. These will be discussed in-depth in section 5.1.3.

For the second experiment, we discussed how we will define the weights for our features when predicting if an examinee is cheating. Section 5.2 discusses our findings concerning this.

5.1 Experiment 1: Gaze Tracking Module

Here we discuss our findings where we try to find a suitable gaze tracking model for our proctoring system. We had to carry out our experiments for two set of models: the iTracker Model [1] and L2cs-net Model [2]. Details about our experiment are discussed below.

5.1.1 iTracker [1]

The table 5.1 below summarises our findings for the iTracker model. The initial iTracker model's implementation was intended for another data set, however we found another work that implemented it for training and testing this model architecture on the MPIIFaceGaze data set [42]. They also had this code available in a public repository [43]. We used their version of the model to conduct our experiments.

System	Epoch	Learning Rate	FPS	Error (cm)
rtx 2060 + r5 2600, 16GB	10		13.6	2.4932
rtx 3060 + core i5 12400, 32GB	10		15.4	2.5865
r5 2600, 16GB	10		12.5	2.6455
core i5 12400, 32 GB	10	0.0001	10.2	2.5357
i3-3300M, 8 GB	10		8.4	2.7664
[42]	25		-	2.3

TABLE 5.1: Comparison of the iTracker model ran with different configurations

We trained the model using the MPIIFaceGaze data set for 10 epochs with a learning rate of 0.0001. The error rate for each system in centimeters and they are the average Euclidian distance of the prediction point from the actual gaze point. The latency, measured in the number of frames processed per second (FPS) is calculated by setting the batch size to 1 and measure the number of outputs the model produces per second.

The first experiment used an R5 2600 processor and an RTX 2060 graphics card with 16Gb of RAM and this system resulting in an fps of 13.6 and an error rate of 2.4932. The second experiment used an i5-6600K processor and an RTX 3060 graphics card, resulting in an fps of 15.4 and an error rate of 2.5865. The third and fourth experiments used only the processors of the first two, but without the graphics cards using the cpu version of PyTorch, and achieved lower FPS as shown in the table above.

The final row of the table provides an "expected value" according to a study [42], which suggests that the expected fps for a model trained for 25 epochs should produce an error rate of approximately 2.3. They, however, did not measure the speed or any metric related to FPS.

5.1.2 L2cs-Net [2]

This model predicted the gaze angle instead of the a Cardasian point. Furthermore, they predicted the horizontal and vertical angle separately. A more detailed summary of the inner architecture of this model can be found in section 4.1.2.2. Their work also included a repository [44] of their implementation where they also inlcuded a pre-trained model. The results of running this pre-trained model on different systems for our experiment are shown below in table 5.2.

System	Epoch	LR	FPS	Vertical Error (Deg)	Horizontal Error (Deg)
rtx 2060 + r5 2600	10		32.4	7.5	7.1
rtx 3060 + i5 6600k	10		61.7	8.2	7.9
r5 2600	-		-	-	-
core i5 6600k	-	0.0001	-	-	-
i3-3300M, 8 GB	-		-	-	-
[2]	50		-	3.96	3.92

TABLE 5.2: Comparison of the L2CS model ran with different configurations

As shown in the table above, the model is only runnable on systems with a GPU. This most likely because ResNet50, the architecture which this model is based on, is computationally very expensive (see appendix section B.7. Therefor, this model, although shows very high results for systems with GPUs (around 32 and 61 FPS), it is not computationally efficient on low-end systems. This makes it unsuitable for our proctoring system as one of our requirements was that it needs to be optimised for all types of systems.

5.1.3 Analysis

The primary objective of our experiments was to find a suitable gaze tracking model for our proctoring system. We had outlined our requirements and conducted a few experiments to compare two gaze tracking models. The second one, L2cs-net had a very low error rate (both vertically and horizontally) and an extremely good FPS rate. However, it failed to run on systems with lower hardware resources.

The iTracker model also produced error rates considerable to the L2cs-net model. The only notable difference was that it was much slower on when testing on high-end systems. However, it did also run on the lower-end systems which the L2cs-net could not do. Furthermore, the iTracker was able to keep the FPS rate almost consist ant (8-15 FPS) across all systems it was tested against.

Because of all these reasons, the iTracker is the superior choice for our proctoring system.

5.2 Experiment 2: Determining the weights of the Unfair Means Features

This portion's objective is to define suitable values for W_1, W_2, W_3 and W_4 for the equation below. Details about the equation have already been discussed when we desgined it in the previous chapter.

$$P = W_1.t + W_2.f + W_3.N_R + W_4.N_H$$

Unfortunately, there are no publicly available datasets where people participate in online exams and try to cheat. Therefor, we are required to create our own dataset by gathering 20 participants and put them in an environment for an online examination. We will then have some of them cheat and label them accordingly. This dataset will be required to perform logistic regression to determine the weights.

Gathering 20 volunteers and conducting a fully simulated online exam will take considerable effort and time. We are opting to do this portion of our experiments in the future with the necesasy resources.

Chapter 6

Conclusion and Future Works

Through extensive research and analysis, we have proposed a design for a comprehensive proctoring system that leverages eye gaze tracking technology to detect suspicious behaviors during examinations.

Our design emphasises on modality, enabling a clear separation of concerns between the overall proctoring system and the gaze tracking aspects. This design brings many aspects to the system like flexibility, scalability and maintainability. By keeping the gaze tracking as a separate module, developers and end-users can make adjustments to the overall proctoring system without having to change the complex end-to-end deep learning model. We have also discussed why this system needs to be installed locally on each examinee's system and have placed great emphasis on optimising the algorithms to ensure that it can run on any system, as students rarely have high-end computers.

In terms of detecting suspicious behaviour, we have highlighted the significance of accurately tracking and interpreting gaze movements within restricted regions. We have discussed how to generate these restricted regions for each individual examinee and have then set up restrictions on this region to further remove false positives.

Furthermore, out research also includes some experiments to determine a suitable algorithm for our gaze tracking model. We have defined our requirements, namely accuracy and speed and have discussed why these are necessary. We then further combined these requirements with the system requiring very optimised algorithms to determine the third requirement. We selected two popular gaze tracking models that are publicly available and tested them on a set of systems with varying hardware. We then analysed our findings and drew conclusions about which of them is more suitable in our scenario.

We also discussed how the features related to unfair means, like the duration or number of times the examinee looks away from the screen, relate to predicting if the examinee is actually cheating. We have also desinged an experiment to determine the weights of each of these features.

In conclusion, the research contributes to the field of academic integrity by presenting a proactive solution to combat exam malpractice through the integration of eye gaze tracking technology into a comprehensive proctoring system.

6.1 Challenges

There are a few challenges for the system we designed. Firstly, the test-taker could simply be staring off into the other direction without any dishonest intentions. A warning issue could be implemented after a certain time restriction, however, other analysis would need to be done if this is too distracting from the exam or if this alert would pop up by any error from the gaze tracking model.

Another issue is that the space between the screen and script. That test-taker can simply put a device there. One option is to mark that zone as warning zone if the proctoring system detects prolonged gaze there. However, this still needs to be verified as it could potentially produce a significant number of false positives.

We only tested two gaze tracking models. More models need to be considered and experimented on to find one much more ideal to our scenario. We also only conducted tests on 5 systems. A much larger set of systems need to be used.

Lastly, we have only designed the second experiment. Due to time constraints, it was difficult to gather 20 participants and conduct an online mock examination. We will have to perform this experiment in the future and determine the weights for the features that we have defined. Afterwards, we will have to conduct a usability study on this system as well. Due to time constraints, these tasks are yet to be fulfilled.

6.2 Future Work

As mentioned in the previous section, more research needs to be carried out on to find a much more suitable gaze tracking model. Their performances also needs to be tested against a larger set of systems.

Besides that, there is also the issue of the space between the screen and the exam script. More research needs to be done to find a suitable solution. Other issues involve incorporating other signs of cheating and how to use gaze tracking to detect them.

Factors such as hardware requirements, user acceptance, and privacy concerns will also require careful consideration and continuous improvement. Further research and development in these areas will contribute to the refinement and effectiveness of our proctoring system.

Besides that, we have mentioned that we are yet to perform our second portion of our experiments. This will involve gathering 20 participants and conducting a mock examination. We will them have to generate our own dataset and use it to determine the suitable weights for each of the features for detecting unfair means. Afterwards, our system will need to undergo usability tests to determine if it is suitable for actual examinations.

Lastly, whenever the issue of monitoring students through video feed is appears, there are always issues with privacy and breach of ethics. Further research in these domains are also necessary to fully present our proctoring system as a working solution for online exams.

Appendix A

A.1 Eye Movements when Reading

According to [37], during reading, the eye makes jerky movements called saccades followed by fixations. Perception occurs during the fixation periods, which account for approximately 94% of the time elapsed. The eye moves backwards over the text as well as forwards, in what are known as regressions. A study carried out in [45] also claims that if the text is complex there will be more regressions as well. This is because more complex words have more data that needs to be retrieved and therefor, more backward movement to provide the human brain with the necessary information.

Research in [7] shows that reading detection can be achieved in two ways. First by analyzing eye movement patterns through looking for fixations, saccades, regression and return-sweeps. These cues are shown in the figure below.

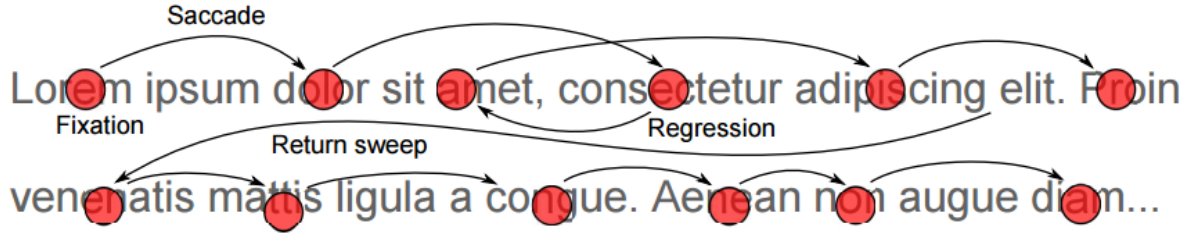

FIGURE A.1: Eye Movement during reading [7]

Now consider if a person is trying to read a piece of text that requires a considerable amount of effort to understand. They will try to re-read it multiple times and hence this will cause their eyes to move backwards over the text.

A.2 Natural Human Viewing Angle

According to [38], the whole human viewing angle is about 180° horizontally and 130° vertically. Each eye's individual field of vision covers only about 120° horizontally and by combining both eyes, a human can cover 180°.

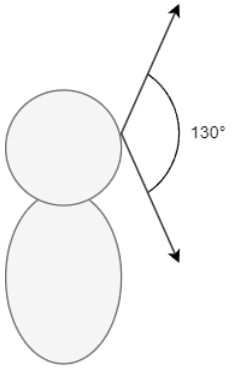

FIGURE A.2: Vertical Viewing Range

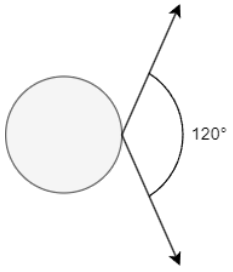

FIGURE A.3: Horizontal Viewing Range

It has been observed that humans will usually tilt their heads to keep the viewing distance similar for both eyes. Thus, to look at anything that is outside of 120° horizontally, they will almost always tilt their heads to do so. Similarly, they will also move their heads to look at an object that exceed the vertical viewing angle of 130°.

Additionally, [38] also states that when a human focus on tasks that requires more visual angle, can be read only by fovea - a part of the eye located in the middle of the Macula on the retina. This fovea's view is around from 1°to 5°from the human view eye.

Appendix B

B.1 Linear Regression

Linear regression is simple yet powerful algorithm used in data science. It uses supervised learning methods to simplify the relationship between variables into a linear form [9]. It is typically used for relationships that are continuous in nature. For example, below is simple relation between an independent variable and a dependent variable.

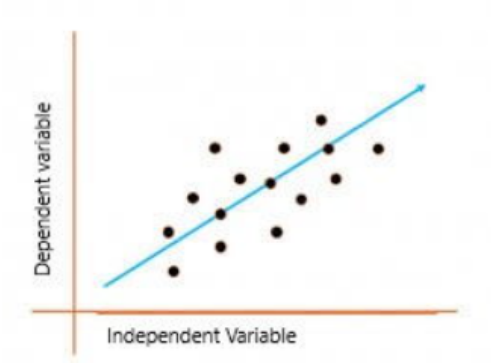

FIGURE B.1: Simple Linear Regression [8]

Linear regression is the process of determining a linear relationship between the two variables. It involves producing a best-fit line through the points given as shown in the figure above.

The line is usually in the form of $Y_i = \beta_0 + \beta_1 X_1$ where where Y_i = Dependent variable, β_0 = constant/Intercept, β_1 = Slope/Intercept, X_i = Independent variable [8]. To produce this best-fit line, we try out different values for β_0 and β_1 and try to minimize the error found. A detailed break down of the terminologies is illustrated in the figure on the next page.

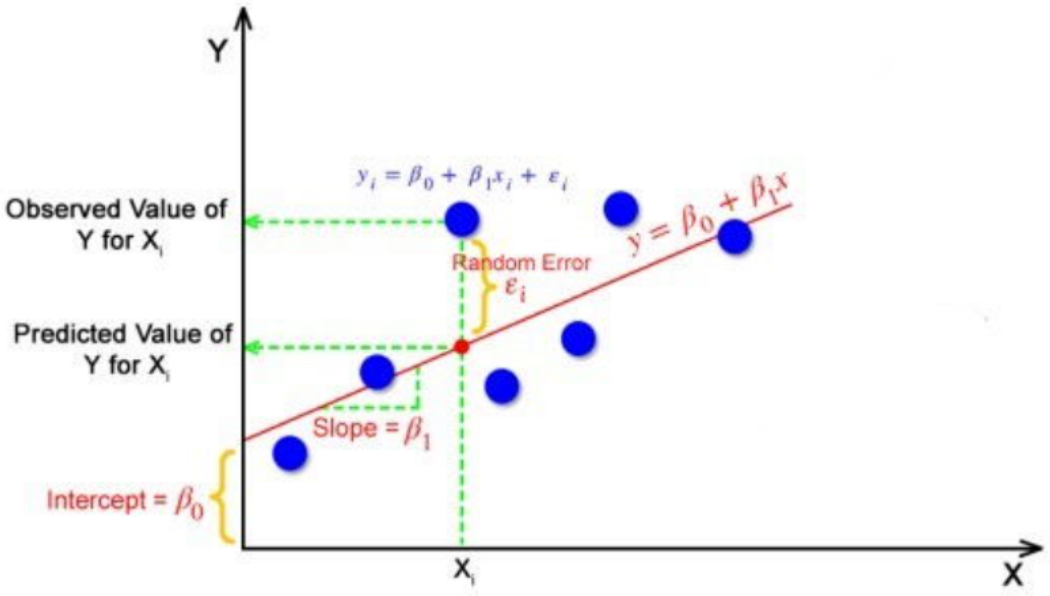

FIGURE B.2: Simple Linear Regression Terms [8]

In regression, the difference between the observed value of the dependent variable(y_i) and the predicted value ($y_{predicted}$) is called the residuals [8, 9]. There are numerous types of error that can be used to find the best fit line. One of the most popular one is the mean squared error (MSE) function. Here, we square the residual, i.e., $(y_{predicted} - y_i)^2$ for each value found for i and then sum them up [8]. The overall equation is as follows.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - y_{predicted})^2$$

Here, N is the total number of points available, and $y_{predicted} = (\beta_1 x_i + \beta_0)$.

Now this MSE is known as a cost function. We will go through values of β_0 and β_1 and try to minimize the value of this cost function. Another popular algorithm for finding the optimal value is Gradient Descent. Here we iteratively update the values of β_0 and β_1 until we get an optimal solution. The graph below shows how this is used to find the optimal value of β_0.

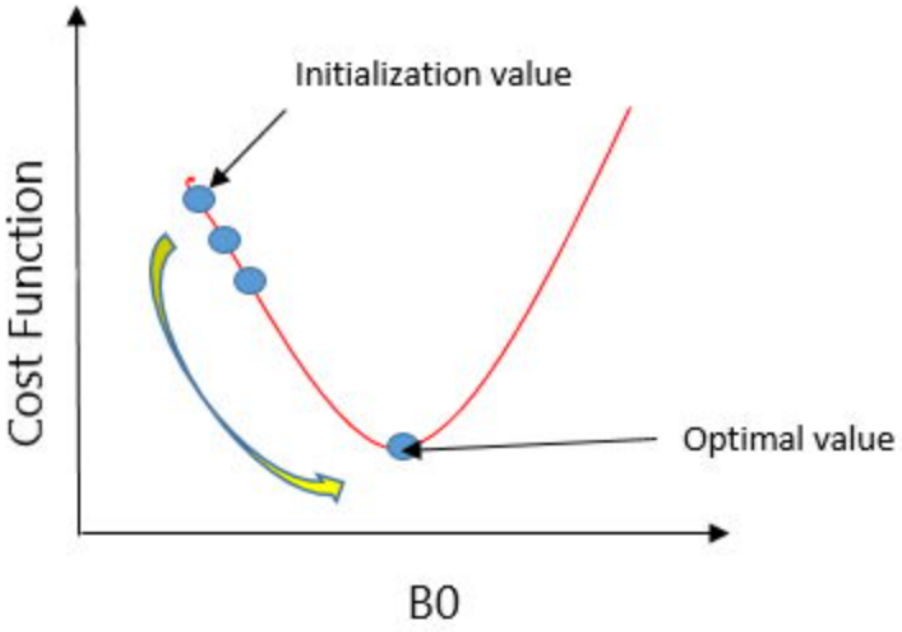

FIGURE B.3: Gradient Descent for β_0 [8]

To update β_0 and β_1 in each iteration, we need use the gradient at that point and a constant α which is known as the learning rate. To find the gradient, we calculate the partial derivatives of the MSE with respective to β_0 and β_1 separately. This is shown below.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)^2$$
$$\frac{\partial MSE}{\partial \beta_0} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)$$
$$\frac{\partial MSE}{\partial \beta_1} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_i - y_i)x_1$$

Now, the values for β_0 and β_1 in the next iteration can be calculated as follows.

$$\beta_0 = \beta_0 - \alpha.\frac{\partial MSE}{\partial \beta_0}$$
$$\beta_1 = \beta_1 - \alpha.\frac{\partial MSE}{\partial \beta_1}$$

Here, α denotes the learning rate. This is like the number of steps the algotithm will take for each iteration. With this, the MSE for both β_0 and β_1 should converge to a minima.

B.2 Logistic Regression

Logistic regression is another powerful supervised ML algorithm used for binary classification problems (when target is categorical) [9]. Therefor, the main difference between this method and linear regression is that this one's range is between 0 and 1. The relationship is defined to have a nonlinear log transformation function as resembling that below.

$$Logistic function = \frac{1}{1 + e^{-x}}$$

Plotting the curve for values between -20 to 20 into the above logistic function produces the following graph.

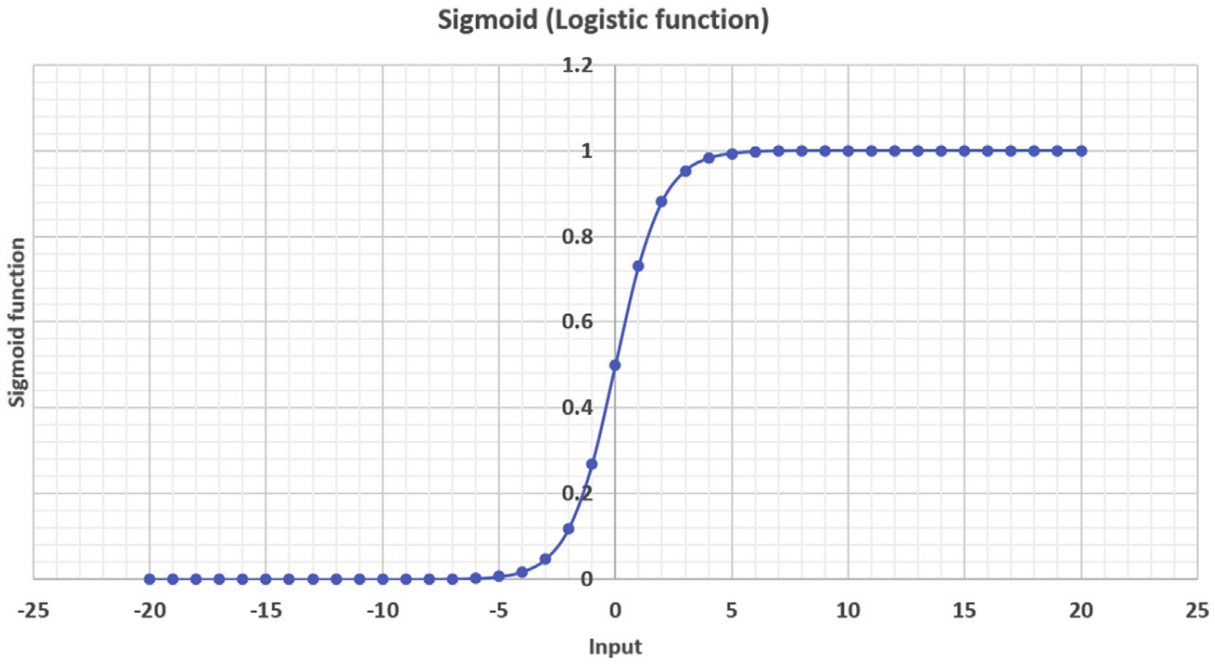

FIGURE B.4: Logistic regression applied to a range of -20 to 20 [9]

To help explain why this is better for categorical relationships, consider if the points on the graph are on two extremes only as the result can only be 1 or 0 as shown on the graph on the next page.

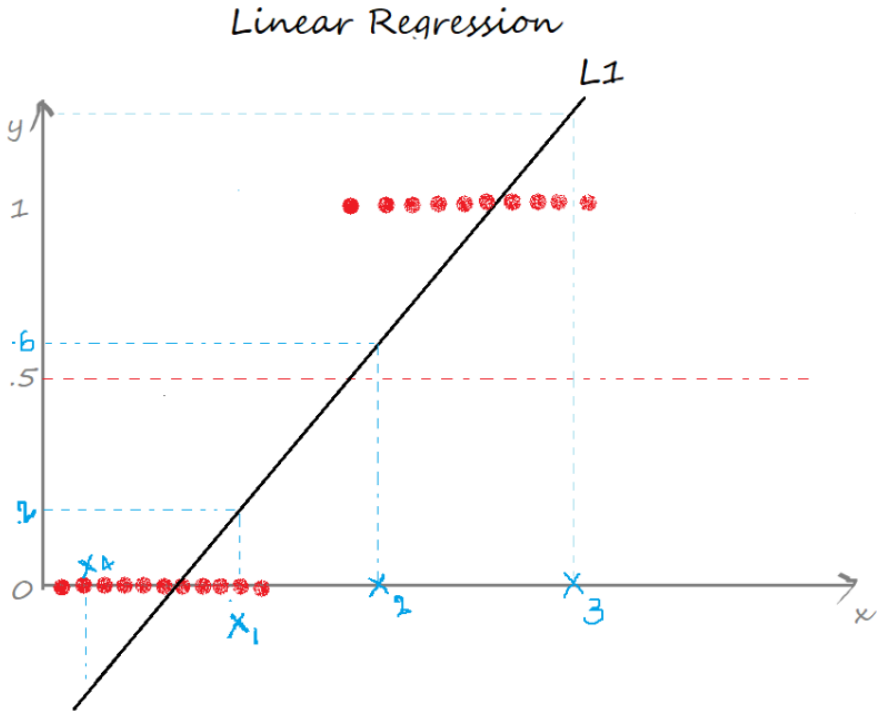

FIGURE B.5: Linear Regression on Categorical Relationship [10]

It is easy to see that that many points to do not come close to the line. A logistic function, however, can be much more suitable when given the right weights [10]. For the converting a linear relationship to a logistic one, we usually denote the linear equation as z and put it inside the logistic equation as shown. Using the linear relationship from the previous section, z can be written as $z = \beta_1 x_i + \beta_0$. This, the logistic function σ becomes

$$\sigma = \frac{1}{1 + e^{-z}}$$
$$\sigma = \frac{1}{1 + e^{-(\beta_1 x_i + \beta_0)}}$$

In logistic regression, as the output is a probability value between 0 or 1, mean squared error would not be the right choice [10]. There are other metrics like cross-entropy loss function [10] or maximum likelihood estimation [9]. We then use the chosen matric and along with a suitable learning rate to converge on the optimal values for β_0 and β_1, similar to the previous section.

B.3 Neural Networks and End-to-End Models [3]

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

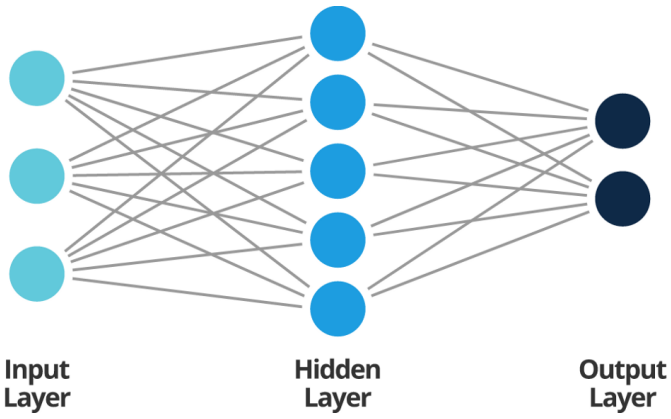

FIGURE B.6: Neural Network

End-to-End models, also known as end-to-end learning or end-to-end systems, refer to machine learning models or systems that aim to directly map the input data to the desired output, without explicitly designing or relying on intermediate steps or modules. In an end-to-end approach, the model learns to perform the entire task in a single unified framework, typically using a deep neural network. This contrasts with traditional approaches that involve manually engineering multiple components or stages to solve a problem. End-to-end models are designed to automatically learn complex representations and internal representations directly from raw data, which can simplify the overall system architecture and potentially improve performance

B.4 Convolutional Neural Networks [4]

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

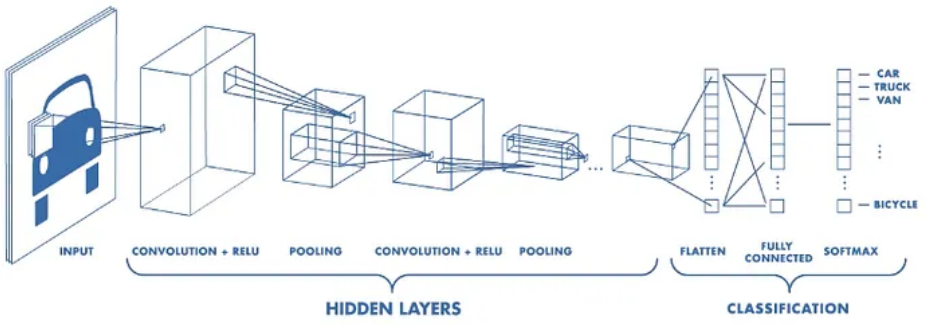

FIGURE B.7: CNN Architecture

Convolution Layer

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

Pooling Layer

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation,

which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

Fully Connected Layer

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output.

B.5 Long Short Term Memory Networks [5]

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data, such as time series, speech, and text. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting.

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period of time. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell.

B.6 Convolutional LSTM

A Convolutional LSTM (Long Short-Term Memory) is an extension of the traditional LSTM architecture that incorporates convolutional operations.

Convolutional LSTMs combine the spatial processing capabilities of convolutional layers with the temporal modeling capabilities of LSTMs. They are particularly useful for tasks that involve both spatial and temporal information, such as video analysis, action recognition, and spatiotemporal prediction.

The key idea behind Convolutional LSTMs is to replace the matrix multiplication operations in the LSTM units with convolutional operations. This allows the network to effectively process inputs with spatial structure, such as images or video frames, while still maintaining the ability to capture temporal dependencies over time.

In a Convolutional LSTM, the input and hidden states of the LSTM units are tensors with multiple channels (e.g., image frames with RGB channels). Convolutional operations are applied to these tensors along both the spatial and temporal dimensions. By convolving the input and hidden states, the network can learn spatial and temporal patterns, respectively.

The Convolutional LSTM architecture typically consists of multiple stacked layers, with each layer containing ConvLSTM units. The lower layers capture low-level spatial and temporal features, while the higher layers learn more abstract and high-level representations.

B.7 ResNet50 Model

The ResNet50 model is a specific variant of the ResNet (Residual Neural Network) architecture. ResNet is a deep convolutional neural network architecture that addresses the problem of vanishing gradients in very deep networks by introducing residual connections.

The "50" in ResNet50 refers to the number of layers in the network, specifically the number of weight layers or parameter layers. ResNet50 has 50 convolutional layers, including convolutional, pooling, and fully connected layers.

The key innovation of ResNet is the residual connection, also known as a skip connection or shortcut connection. In traditional deep neural networks, each layer learns a set of transformations from the input to the output. However, as the network gets deeper, it becomes challenging for the network to learn these transformations effectively, resulting in degraded performance.

ResNet addresses this issue by introducing skip connections that bypass some layers. These skip connections enable the network to learn residual mappings, i.e., the difference between the desired output and the current output of the layer. By propagating these residuals through the network, the gradients have a shorter path to flow, alleviating the vanishing gradient problem and enabling the network to be trained more effectively.

ResNet50 architecture specifically consists of a series of convolutional blocks. Each block typically consists of several convolutional layers, followed by a shortcut connection. The shortcut connection merges the input of the block with the output of the block, allowing the network to learn the residual mapping. ResNet50 also incorporates identity shortcuts, where the input is directly connected to the output without any additional transformation, to preserve information flow.

Bibliography

[1] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2176–2184.

[2] A. A. Abdelrahman, T. Hempel, A. Khalifa, and A. Al-Hamadi, "L2cs-net: Fine-grained gaze estimation in unconstrained environments," *arXiv preprint arXiv:2203.03339*, 2022.

[3] N. Bressler. (2022) What are neural networks? [Online]. Available: https://www.ibm.com/topics/neural-networks

[4] M. Mishra. (2020) Convolutional neural networks, explained. [Online]. Available: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[5] aditianu1998. (2023) Understanding of lstm networks. [Online]. Available: https://www.geeksforgeeks.org/understanding-of-lstm-networks/

[6] C. Thombare, K. Sapate, A. Rane, and A. Hutke, "Proctoring system," *Journal homepage: www. ijrpr. com ISSN*, vol. 2582, p. 7421.

[7] M. Makame, "Towards assured informed consent in privacy notice design: An eye movement detection approach," Ph.D. dissertation, 06 2016.

[8] K. Mali. (2023) What is linear regression? [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/

[9] H. Belyadi and A. Haghighat, "Chapter 5 - supervised learning," in *Machine Learning Guide for Oil and Gas Using Python*, H. Belyadi and A. Haghighat, Eds. Gulf Professional Publishing, 2021, pp. 169–295. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128219294000044

[10] H. Bonthu. (2021) An introduction to logistic regression. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-logistic-regression/

[11] R. Kareem kadthim and Z. H. Ali, "Survey: Cheating detection in online exams," *International Journal of Engineering Research and Advanced Technology*, vol. 08, no. 01, p. 01–05, 2022.

[12] A. Balderas and J. A. Caballero-Hernández, "Analysis of learning records to detect student cheating on online exams: Case study during covid-19 pandemic," in *Eighth international conference on technological ecosystems for enhancing multiculturality*, 2020, pp. 752–757.

[13] O. R. Harmon and J. Lambrinos, "Are online exams an invitation to cheat?" *The Journal of Economic Education*, vol. 39, no. 2, pp. 116–125, 2008.

[14] R. Raman, H. Vachharajani, P. Nedungadi *et al.*, "Adoption of online proctored examinations by university students during covid-19: Innovation diffusion study," *Education and information technologies*, vol. 26, no. 6, pp. 7339–7358, 2021.

[15] R. M. Al_airaji, I. A. Aljazaery, H. T. Alrikabi, and A. H. M. Alaidi, "Automated cheating detection based on video surveillance in the examination classes," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 16, no. 08, p. pp. 124–137, 04 2022. [Online]. Available: https://online-journals.org/index.php/i-jim/article/view/30157

[16] A. Lee-Post and H. Hapke, "Online learning integrity approaches: Current practices and future solutions." *Online Learning*, vol. 21, no. 1, pp. 135–145, 2017.

[17] S. Kaddoura and A. Gumaei, "Towards effective and efficient online exam systems using deep learning-based cheating detection approach," *Intelligent Systems with Applications*, vol. 16, p. 200153, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305322000904

[18] T. Potluri, "An automated online proctoring system using attentive-net to assess student mischievous behavior," *Multimedia Tools and Applications*, pp. 1–30, 2023.

[19] S. Govind, "Webcam based eye-gaze estimation," *International Journal of Engineering Applied Sciences and Technology*, 2019.

[20] Y.-T. Lin, R.-Y. Lin, Y.-C. Lin, and G. C. Lee, "Real-time eye-gaze estimation using a low-resolution webcam," *Multimedia tools and applications*, vol. 65, no. 3, pp. 543–568, 2013.

[21] N. H. Cuong and H. T. Hoang, "Eye-gaze detection with a single webcam based on geometry features extraction," in *2010 11th International Conference on Control Automation Robotics & Vision*, 2010, pp. 2507–2512.

[22] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proceedings of*

the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, ser. ETRA '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 131–138. [Online]. Available: https://doi.org/10.1145/2857491.2857492

[23] A. Gudi, X. Li, and J. van Gemert, "Efficiency in real-time webcam gaze tracking," in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds. Cham: Springer International Publishing, 2020, pp. 529–543.

[24] S. Park, E. Aksan, X. Zhang, and O. Hilliges, "Towards end-to-end video-based eye-tracking," in *European Conference on Computer Vision (ECCV)*, 2020.

[25] H. Khachatryan and A. L. Rihn. (2017) Eye-tracking methodology and applications in consumer research. [Online]. Available: https://edis.ifas.ufl.edu/publication/FE947

[26] Y. R. Pratama, S. Atin, and I. Afrianto, "Predicting student interests against laptop specifications through application of data mining using c4.5 algorithms," *IOP Conference Series: Materials Science and Engineering*, vol. 662, no. 2, p. 022129, 11 2019. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/662/2/022129

[27] A. Nigam, R. Pasricha, T. Singh, and P. Churi, "A systematic review on ai-based proctoring systems: Past, present and future," *Education and Information Technologies*, vol. 26, no. 5, pp. 6421–6445, 2021.

[28] F. Klijn, M. Mdaghri Alaoui, and M. Vorsatz, "Academic integrity in on-line exams: Evidence from a randomized field experiment," *Journal of Economic Psychology*, vol. 93, p. 102555, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167487022000666

[29] B. Burgess, A. Ginsberg, E. W. Felten, and S. Cohney, "Watching the watchers: bias and vulnerability in remote proctoring software," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 571–588.

[30] M. F. Ansari, P. Kasprowski, and M. Obetkal, "Gaze tracking using an unmodified web camera and convolutional neural network," *Applied Sciences*, vol. 11, no. 19, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/19/9068

[31] N. Dilini, A. Senaratne, T. Yasarathna, N. Warnajith, and L. Seneviratne, "Cheating detection in browser-based online exams through eye gaze tracking," 12 2021, pp. 1–8.

[32] M.-T. Vo and S. G. Kong, "Enhanced gaze tracking using convolutional long short-term memory networks," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 22, no. 2, pp. 117–127, 2022.

[33] Y.-m. Cheung and Q. Peng, "Eye gaze tracking with a web camera in a desktop environment," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 419–430, 2015.

[34] X. Zhou, J. Lin, Z. Zhang, Z. Shao, S. Chen, and H. Liu, "Improved itracker combined with bidirectional long short-term memory for 3d gaze estimation using appearance cues," *Neurocomputing*, vol. 390, pp. 217–225, 2020.

[35] B.-J. Hwang, H.-H. Chen, C.-H. Hsieh, and D.-Y. Huang, "Gaze tracking based on concatenating spatial-temporal features," *Sensors*, vol. 22, no. 2, p. 545, 2022.

[36] N. Aunsri and S. Rattarom, "Novel eye-based features for head pose-free gaze estimation with web camera: New model and low-cost device," *Ain Shams Engineering Journal*, vol. 13, no. 5, p. 101731, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447922000429

[37] A. Dix, *Human-computer Interaction*. Prentice Hall Europe, 1998. [Online]. Available: https://books.google.com.bd/books?id=tNxQAAAAMAAJ

[38] M. Kowalik, "Do-it-yourself eye tracker: impact of the viewing angle on the eye tracking accuracy," *Proceedings of CESCG*, pp. 1–7, 2011.

[39] N. Bressler. (2022) MS Windows NT kernel description. [Online]. Available: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model

[40] K. A. Funes Mora, F. Monay, and J.-M. Odobez, "Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ser. ETRA '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 255–258. [Online]. Available: https://doi.org/10.1145/2578153.2578190

[41] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "It's written all over your face: Full-face appearance-based gaze estimation," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 2299–2308.

[42] Y. Cheng, H. Wang, Y. Bao, and F. Lu, "Appearance-based gaze estimation with deep learning: A review and benchmark," *arXiv preprint arXiv:2104.12668*, 2021.

[43] Y. Cheng, "itracker implementation on mpiigaze," https://github.com/yihuacheng/Itracker, 2021.

[44] A. A.Abdelrahman, "L2cs-net," https://github.com/Ahmednull/L2CS-Net, 2022.

[45] R. Booth and U. Weger, "The function of regressions in reading: Backward eye movements allow rereading," *Memory cognition*, vol. 41, 08 2012.