An automated proctoring assistant in online exams using computer vision

Final Year Project Final Report

A 4th Year Student Name

Nguyen Khanh Luan

Pham Thi Thu Ha

Instructor

Dr. Phan Duy Hung

Bachelor of Computer Science

Hoa Lac campus - FPT University

April 2022

Acknowledgment

We thank our dedicated, supportive instructor, Dr. Phan Duy Hung, for giving us the best guidance and lessons to complete our final graduation project.

We would like to thank University FPT, for giving us the environment to learn and grow both professionally and personally over the years.

We would like to thank all our classmates in Computer Science Major Class, for the collaboration and the support in periods of struggle and difficulty.

Abstract

Cheating or attempting to cheat in education has had the opportunity to become numerous and complex since the outbreak of the COVID-19 pandemic, with teaching and testing conducted online. The learners have easier access to prohibited materials, and it is easier to avoid contact with a human proctor. Such problems raise the need for an automated intelligent proctoring system to help the teacher supervise students.

Therefore, this work proposes a system that can automatically examine students' behaviors through two main cameras. The first camera takes images of a student's frontal face and uses them as input for the facial landmark model, detecting anomalies in a student's face movement. The second camera captures the student's whole body and the surrounding environment, and by using a trained pose recognition model, the system can efficiently classify student actions as suspicious or not. In addition, an object recognition model is also applied simultaneously to detect people and objects appearing through the second camera. Results of this research show good remarks and can be applied in schools, universities experimentally in the future.

Keywords: Computer Vision, Anomaly Detection, Proctoring.

Table of contents

Acknowledgments **3**

Abstract **4**

Table of contents **5**

List of figures **6**

1. Introduction **8**
1.1. Problem and Motivation 8
1.2. Related works 9
1.3. Contribution 10
1.4. Outline 12

2. Methodology **13**
2.1. Problem assumptions 13
2.2. Supervising front camera 13
2.2.1. Eyes tracking 13
2.2.2. Head pose estimation 14
2.2.3. Mouth and hand tracking 14
2.3. Supervising side camera 15
2.3.1. Pose Estimation and detect suspicious behaviors 15
2.3.2. Object detection and people counting 19

3. Experimental results and discussions **20**
3.1. Front camera results 20
3.2. Side camera results 21

4. Conclusion and future works **24**
4.1. Conclusion 24
4.2. Future works 24

Reference **26**

Appendix

List of figures

1	Take the remote test on Safe Exam Browser - a secure browser…….....	8
2	Arrangement of two cameras to capture students' images……	9
3	ID verification…….	10
4	360 degree camera for supervision..…….	11
5	Proposed system…….…….	12
6	Iris tracking approach……..	13
7	Face mesh Mediapipe and PnP OpenCV…….	14
8	Angles represented by different axis rotation……..	14
9	Mediapipe mouth and hand tracking…….	15
10	Action classifier methodology…….	15
11	Recording and labeling video dataset……...	16
12	Collect pose landmarks using Mediapipe…….	16
13	3D landmarks coordinates and student elbow and shoulder angels extraction……..	17
14	Landmarks data extraction from all videos……..	17
15	LSTM Network architecture……	18
16	LSTM network architecture for classifying action……....	19
17	YOLOv4 network structure……..	20
18	Eyes tracking result……..	20
19	Head pose estimation……..	21

20 Mouth tracking and face occlusion detected……. 21

21 Model accuracy and loss on training dataset…… 22

22 Confusion matrix to evaluate model accuracy…….. 22

23 No suspicious action detected……. 23

24 Suspicious action detected…….. 23

25 Results of object detection and person counting……. 23

26 Video logged by the system……. 24

1. Introduction

1.1 Problem and Motivation

Amid the COVID-19 pandemic, most schools and colleges have turned to online learning and testing in response to the long-term quarantine. This form is considered to have many advantages, and after the pandemic, online and offline teaching and learning will be expected to be carried out in parallel with appropriate duration and subjects. Student assessment is an essential part of education. So it requires close supervision to ensure fair results. Offline exam monitoring relies heavily on proctors, who directly monitor students' test results, detecting and recording unusual behaviors. With the online exam, many methods have been implemented, such as monitoring candidates via Zoom online video and audio [1], studying online and taking the test offline under the supervision of a proctor, or taking the online exam on a secure browser [2].

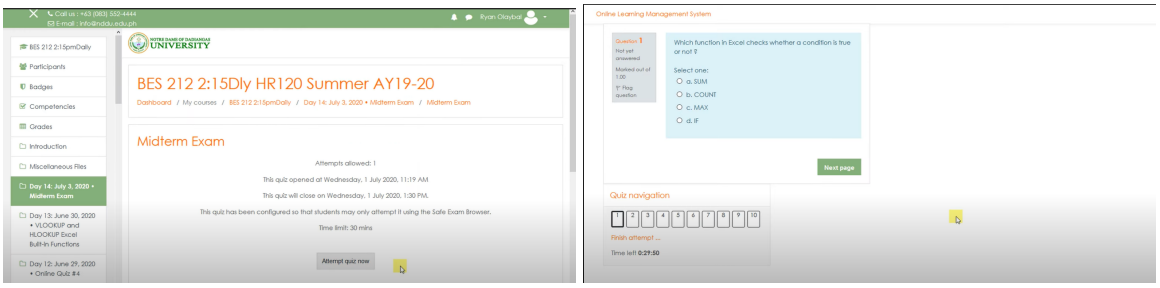
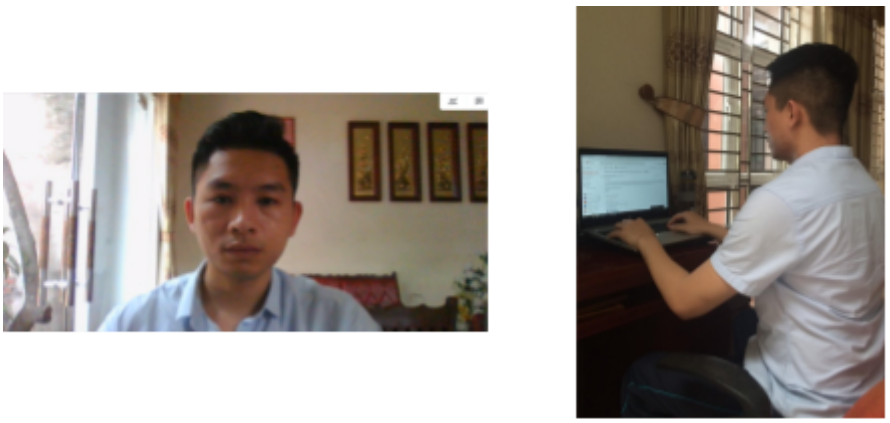


Figure 1. *Take the remote test on Safe Exam Browser - a secure browser*

Most learners today are capable of equipping personal computers and mobile phones. Therefore, a reliable method is being implemented by schools that require students to set up two cameras during the exam: The first camera allows observing the front of a student, the second camera can show the surrounding environment (Fig. 2) [3].



(a) First camera (b) Second camera

Figure 2. *Arrangement of two cameras to capture students' images.*

However, this method still needs the supervision of the supervisors via the cameras. The proctor has to monitor two cameras per student, and it consumes a lot of effort and concentration, which also requires more proctors to observe a larger number of students. The cost of this problem can be estimated from a specific case of FPT University in Hanoi, Vietnam. For example, in the Fall 2021 semester of the Hoa Lac campus, the university organized online exams for 116 subjects. For the JDP113 subject alone, about 1020 students were taking the test. If each online exam room has at least 20 students, 51 online exam rooms are needed. If each exam room has only one proctor, each proctor has to simultaneously observe about 40 screens through the cameras, which does not seem to guarantee the effectiveness of proctoring. So, if two proctors supervise each online exam room, then 102 proctors are needed for just one JDP113 subject. Not to mention all three campuses holding this subject exam and taking other subjects simultaneously, the number of supervisors required will be massive. As such, organizing online exams will require a massive amount of effort, time, and money to ensure the fairness of the assessment. Therefore, the need to apply high technologies and artificial intelligence to solve this problem is critical. This work implements a computer vision system to observe and record unusual behaviors to assist online exam proctoring and reduce the proctor's effort.

1.2. Related works

Several methods adopted recently show high guarantees of test integrity. The following sections discuss some of the anti-cheating solutions to fraudulent attempts.

Søgaard in [4] gives a comprehensive evaluation of Safe Exam Browser as software that ensures a secured environment during a digital Bring-Your-Own-Device

exam. This software will host a user system and shut down unrelated apps or websites. Only predefined apps and tools are allowed to run. It can also disable screen recordings and projection, which leads to test information leakage. However, this method is not as effective as participants can still access outside materials to find answers.

According to an online survey [5], every 1 in 3 students uses mobile phones to cheat during online exams. Bedford et al. introduce the technology that detects devices under usage, searches for similar test contents, and flags certain activities if seen to ensure that the student does not look for answers online [6]. It can also scan the internet, block the sites that are browsed for answers and report them. Nevertheless, as there is no live or auto-proctoring involved, participants can still attempt to cheat by using offline resources.

To avoid fake exam takers or impersonators, Sahil et al. in [7] have devised a system that asks students to provide their ID, photo, and personal information during the registration process. Their data is stored in a database and will then be used to verify legitimate candidates using face recognition and other tools. Nevertheless, this method still needs multiple checks such as biometrics, keystroke authentication and cross-questions to become effective.

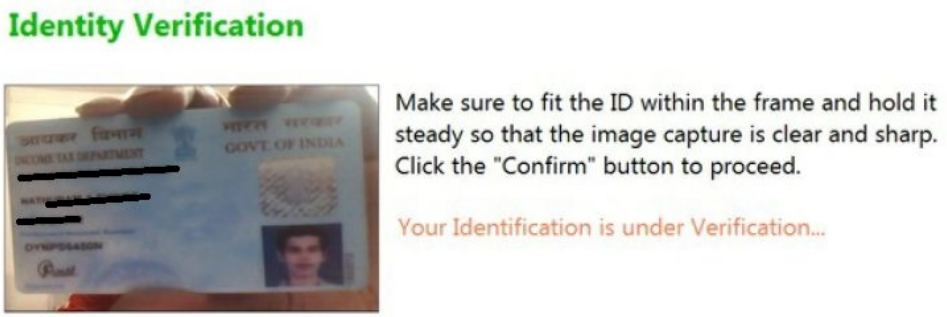


Figure 3. *ID verification*

Record and review proctoring is another method that shows high performance in capturing malpractice in online exams. Authors in the paper [8] have proposed an auto-proctoring system that can run by default and generate reports of anomalous instances. An expert team later uses this to check in detail and verify the integrity of test-takers. Although this method is cheaper than live proctoring, it can take a lot of time.

Turani et al. propose a proctoring system using a 360-degree security camera that can flexibly capture images in the surrounding environment [9]. These cameras can be attached to a computer screen or headgear to track a candidate's gaze and actions in a room using machine learning algorithms. This device provides excellent audio and

video qualities but is expensive and might be cumbersome for some students to wear on the head.

Figure 4. *360-degree camera for supervision*

Coming up with solutions against online exam cheating during the COVID-19 pandemic is currently a hot field of research. However, existing methodologies still exhibit several limits. Although some approaches show remarkable performance to deceitful behavior, as in the case of mobile phone prevention or online ID authentication, they are not anti-cheating stand-alone solutions and do not have any auto-proctoring process. Others are too expensive or too time-consuming such as using 360 cameras and record-review methods. Research in this area is expanding, and researchers are continuously experimenting and proposing solutions to resolve it.

1.3. Contribution

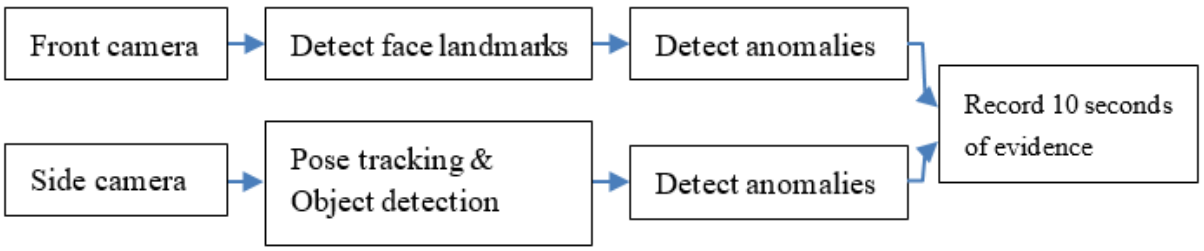


Figure 5. *Proposed system.*

Upon reviewing the above solutions and analyzing the constraints, this study proposes an automated AI-based proctoring system to help teachers monitor and supervise students while taking online exams using computer vision (Fig. 5). The input images are fed from two cameras. With the front camera, the work uses a face landmark model [10] to track suspicious head, eyes, lips movement and, after that, detect abnormality. A pose recognition model based on a long short-term memory (LSTM) [11] to classify malpractice behavior with the side camera . In addition, this study uses object detection to prevent the use of prohibited materials such as phones, electronic devices and books. This research is being experimented for online exams in the university.

1.4. Outline

In this thesis, we address the problem of remote proctoring, specifically:

Section 1 gives an introduction about the problem, motivation, and related works about remote proctoring.

Section 2 presents our proposed model for remote proctoring through two cameras with specific assumptions as well as methodologies to detect facial gestures and behavior along with the model to recognize people and objects.

Section 3 showed some experimental results and evaluation of each model in detecting facial gestures, behaviors, and objects.

Section 4 concludes the thesis and then makes some future work on the subject.

The final section is the list of all reference works helping to create this thesis.

2. Methodology

In this thesis, an automated proctoring assistant is proposed to aid official proctors in supervising students during an online exam.

2.1. Problem assumptions

In this research, certain assumptions are made for taking examination:

- Students need to sit in front of the computer during the test-taking; other moving or cheating behavior are not allowed.

- Cameras are set up to clearly see the faces of test-takers, their bodies, and the surrounding environments.
- Students are not allowed to bring electronic devices that can receive or transmit information, such as mobile phones, USBs, and memory cards.
- Students can only use pens, white paper, or material approved by the official proctor.
- Other than the test taker, no one is allowed to enter the room, including friends or family members.

2.2. Supervising front camera

This work uses Mediapipe, a fast and accurate framework that offers machine learning solutions like facial landmarks detection, hand detection, and pose estimation [12].

2.2.1. Eyes tracking

Using landmarks coordinates output from Mediapipe Iris detection model [13], eye ratio is calculated as the distance from the iris to the right outermost eyes on the other. A suspicious glance is counted as the ratio is greater than one-third, suggesting that the student is peeking to the far right or far left from the front camera.

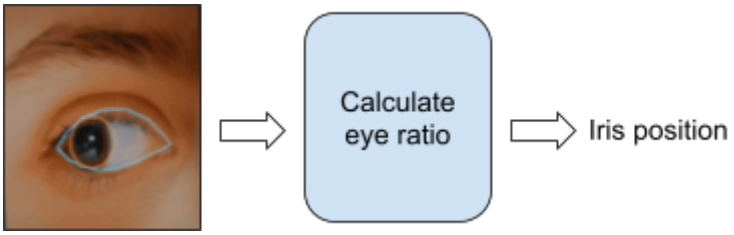


Figure 6. *Iris tracking approach*

From extracted eye landmark coordinates, eye width and distance from the iris to the eye edge are calculated to determine the relative position of an iris.

2.2.2. Head pose estimation

Using a combination of Mediapipe 3D face mesh [10] and OpenCV Perspective-n-Point (PnP) [14] is to estimate head direction from 2D image feeds. Given a set of 3D points in the world and their corresponding 2D projections in the image, PnP can estimate the pose of a calibrated camera. The output of the PnP

solution is a rotation matrix, which is then converted to angles to catch the student's head movement, such as turning right, left, or down.

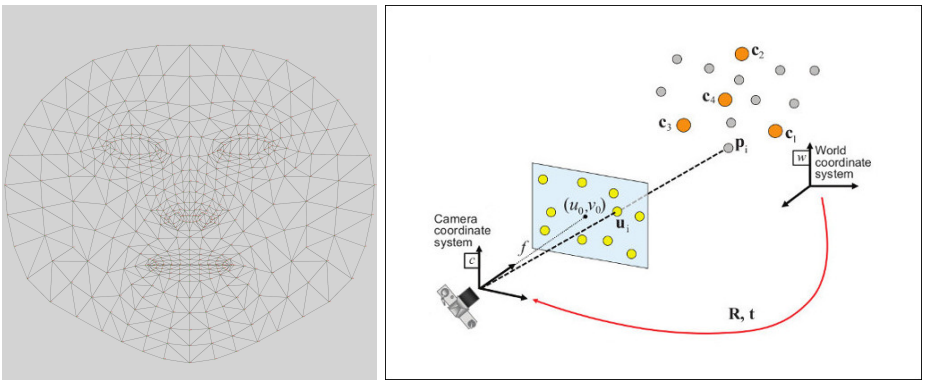


Figure 7. *Face mesh Mediapipe and PnP OpenCV*

Because the Mediapipe Face Mesh can give 3D landmarks coordinates, namely the width, height, and depth of a landmark point in an image frame. Feed this input to the OpenCV PnP method can produce accurate head pose angles.

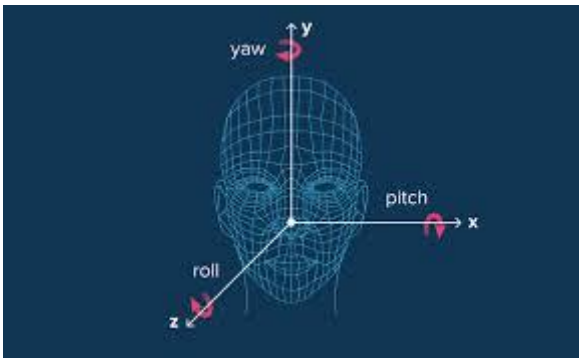


Figure 8. *Angles are represented by different axis rotation*

2.2.3. Mouth and hand tracking

To prevent talking or efforts trying to talk, this work suggests a fusion of the hands landmarks [15] and the mouth landmarks. A student is considered talking when the distance between his or her lips is greater than a predefined distance and over a period of time. Efforts to occlude the mouth using hands are also dealt with by making sure mouth coordinates do not fall into hand area coordinates.

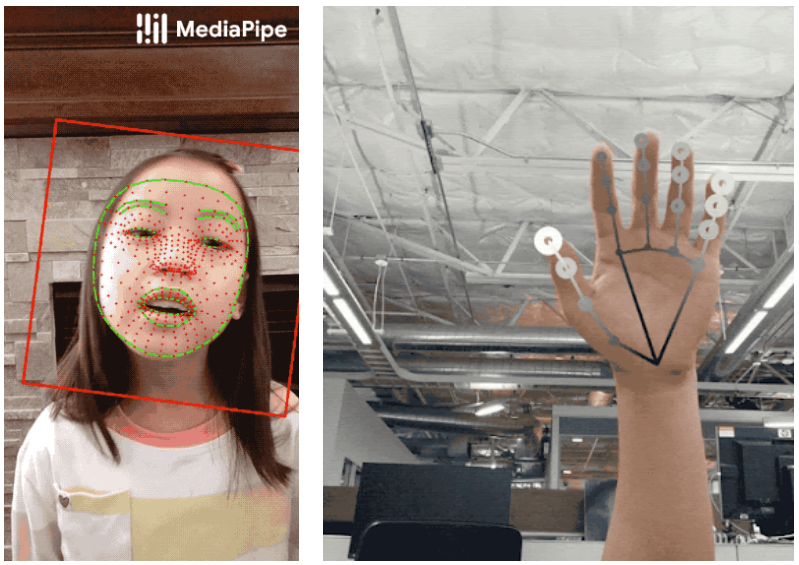


Figure 9 . *Mediapipe mouth and hand tracking*

2.3. Supervising side camera

This thesis proposes a binary classifier for student action and uses the YOLOv4 object detector [16] to detect the cheating attempts of a student, such as reaching for prohibited materials, using a mobile phone, or receiving help from other people.

2.3.1. Pose tracking and detecting suspicious actions

a. Methodology

This research makes use of a sequential model to efficiently classify student activities during the exam. Specifically, an LSTM model is utilized as it will predict based on 30 continuous image frames (1-second duration) which record student movement, not just from a single frame. With vast numbers of these sequence data, the model will quickly learn the pattern and classify student actions as suspicious or non-suspicious.

Figure 10. *The action classifier methodology*

uses a pose detector on the video to get landmarks coordinates for each frame, then feeds this landmarks data to the LSTM network to predict the behavior of a student.

b. Data collection

To make the pose detector learn to classify a student's action, an extensive amount of data needs to be collected. Fortunately, OpenCV helps automate this process of recording and labeling in this work. Since the model calculates the probability of student action as cheating or not, this research records instances when a student is performing suspicious actions and when a student is doing the test. A batch of 720 videos of 1 second long is then cut and labeled as cheating and also with the other 720 videos which are labeled as non-cheating.

Figure 11. *Recording and labeling video dataset*

Next, this work extracts student pose landmarks from these videos using Mediapipe Pose [17]. These pose landmarks are the width, height, and depth of corresponding points in camera frame coordinates and are lightweight enough to store in the computer memory. They are ready to be used as a training dataset for our pose recognition model but to increase model robustness, this work also adds calculated joint angles of the student's shoulders and elbows to detect any suspicious actions.

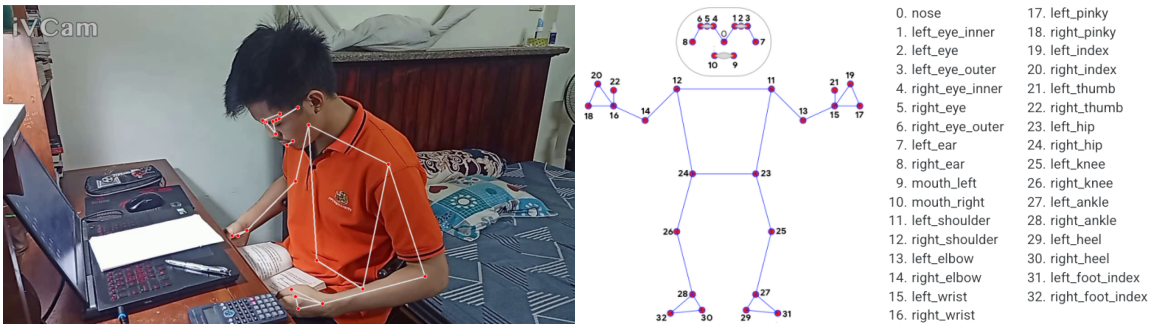


Figure 12. *An example of the pose landmarks*

Because the model heavily relies on pose landmark coordinates, without considering color information in the environment. An extensive extraction of landmark coordinates will be needed. This work extracts 25 upper body landmarks, each landmark contains 4 fields of information namely x, y, z, visibility corresponding to width, height, depth and probability of the landmark in the camera frame. The Mediapipe pose can also return the mimic of 3D real-world coordinates that represent the relative position of a landmark to one another. To increase model robustness, this work also calculates 4 angles that represent a student's left and right shoulder and elbow using simple 2d width and height landmark coordinates. Total extraction gives 204 fields of landmark data per frame.

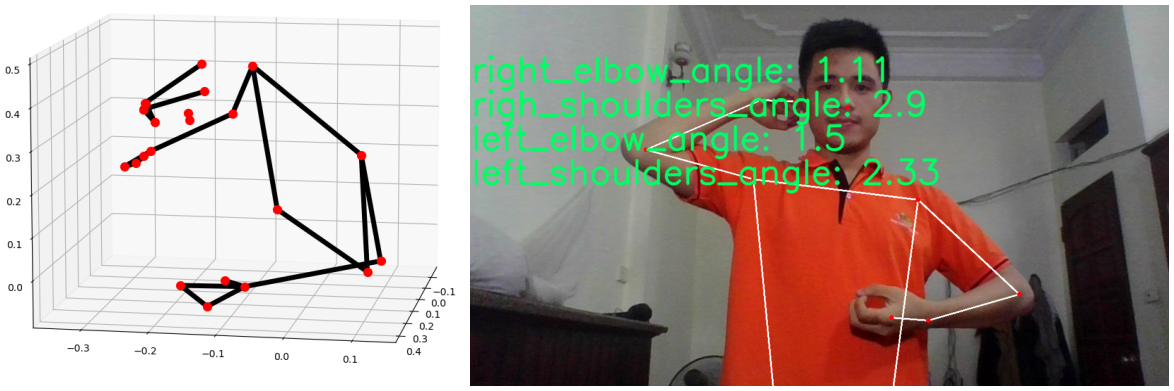


Figure 13. *3D landmarks coordinates and angles extractions of student's elbow and shoulder*

The last step in data collection is to pull these 204 landmark data out of 1440 videos (30 frames duration) and use them for the action classifier. Simply iterating through all videos and extracting mentioned landmarks gives a final dataset of shape (1440, 30, 204) NumPy array.

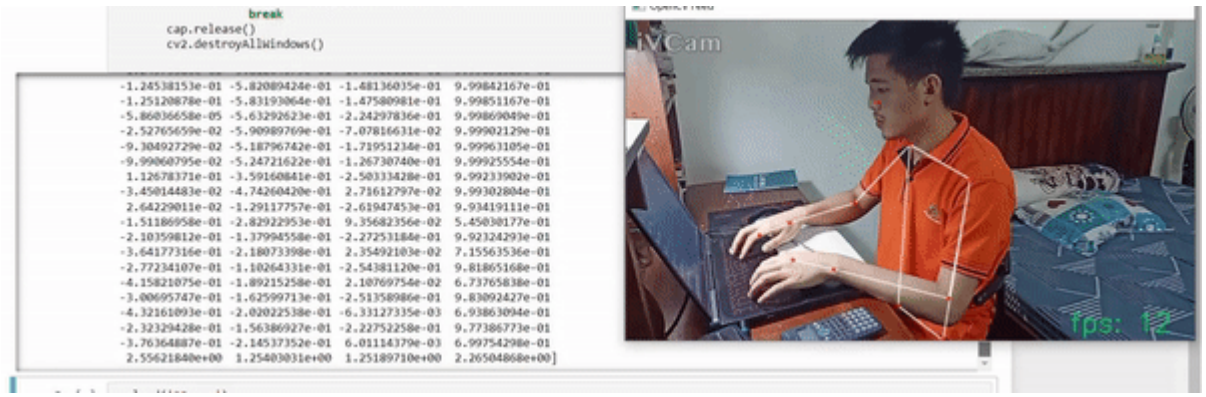


Figure 14. *Landmarks data extraction from video dataset*

c. Pose recognition model

An LSTM neural network is designed specifically for data sequences as it takes into consideration all previous inputs to generalize an output. Because an LSTM cell can remember the context for long input sequences, this network is capable of solving problems involving sequential data like language translation, music composition, or action recognition.

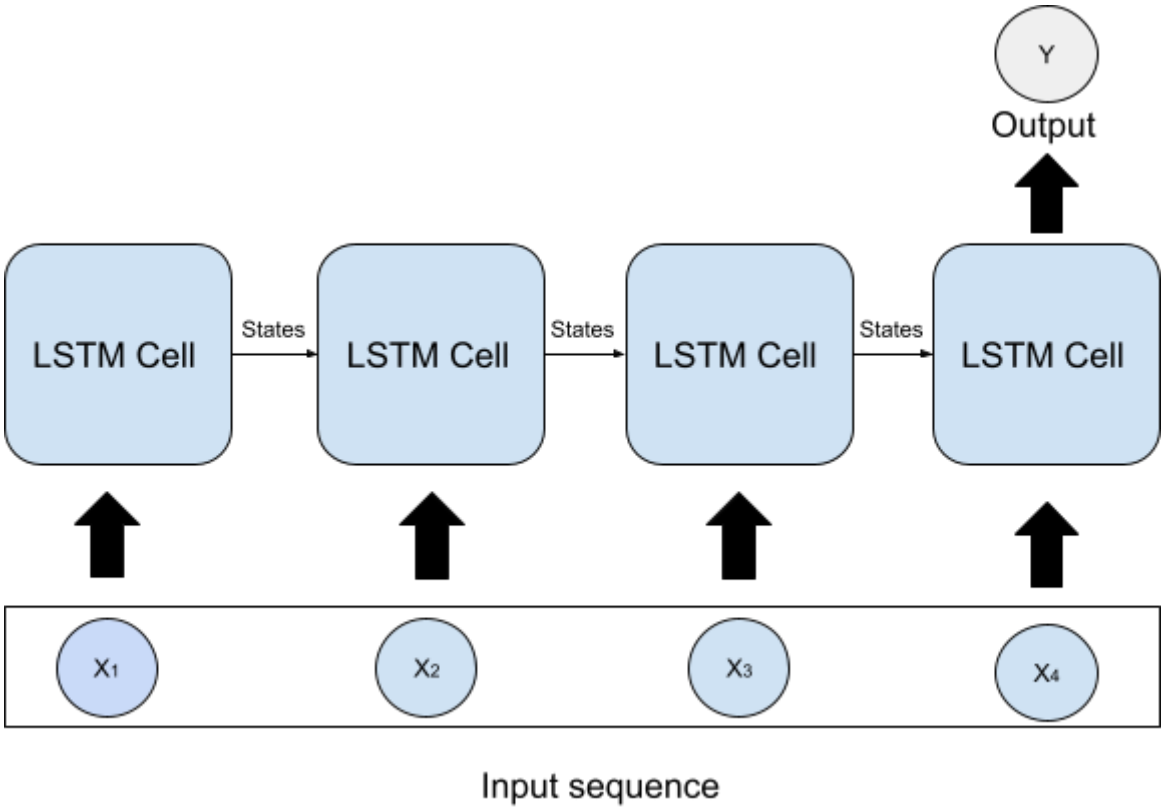


Figure 15. *LSTM Network architecture*

The LSTM network architecture used in this work comprises 3 LSTM connected layers and 3 dense layers (Fig 16), the activation function for this network is a

rectified linear unit. With the data input shape of (30, 204), meaning 30 continuous frames of 204 landmarks data, the model will predict the output of shape (2,) which is the probability of cheating and non-cheating labels.

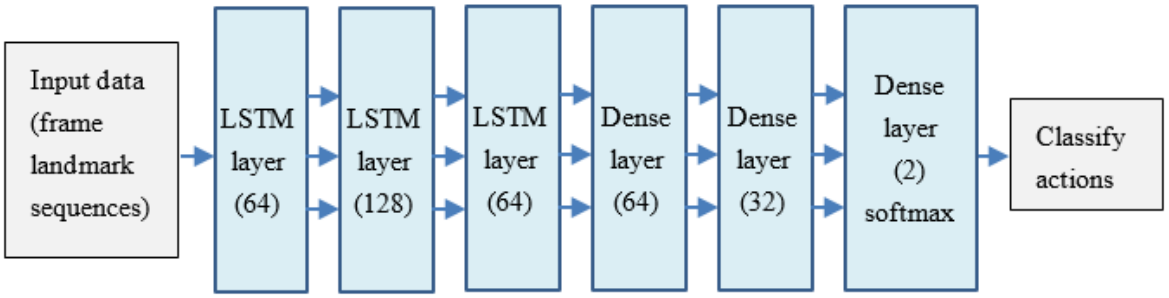


Figure 16. *LSTM network architecture for classifying action.*

2.3.2. Object detection and people counting

YOLOv4 is applied to this work for the purpose of identifying people and objects. It is a CNN network model created from the combination of convolutional layers and connected layers. In which the convolutional layers will extract the features of the image, and the full-connected layers will predict the probability and the coordinates of the object. As a result, the model can detect and count the number of people and objects allowed to appear through the camera.

YOLOv4 mainly consists of 3 parts: backbone, neck and head. The authors used CSPDarknet53 for the model's backbone, which augments the learning capacity of CNN. The neck part is used for SPP and PAN, YOLOv3 is the head of the model. In addition, Bag of Freebies and Bag of Specials are also added as backbone and detector to improve performance such as cost function, and accuracy.

The custom model applies a pre-trained model with Alex's Darknet to recognize 8 types of objects for this specific case - 8 classes including: person, tv monitor, laptop, mouse, remote, keyboard, cell phone, book. The model will detect, draw predicted bounding boxes and count the number of objects of each class with confidence.

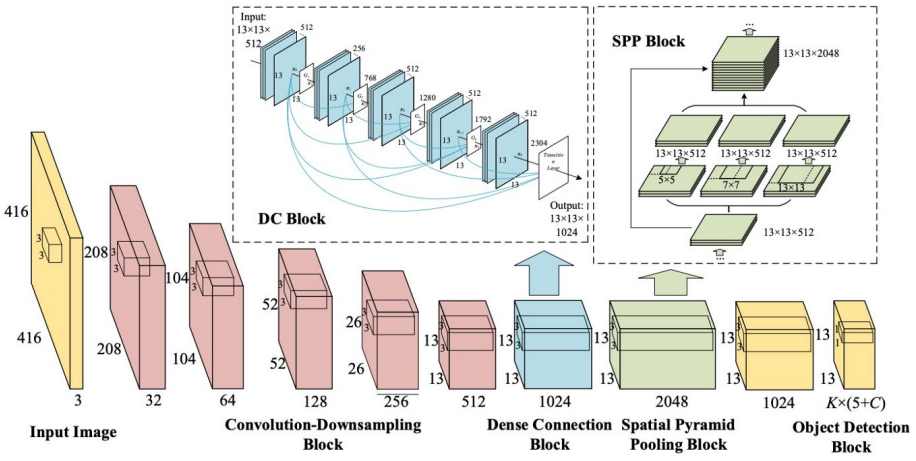


Figure 17. *YOLOv4 network structure*

3. Experimental results and discussions

An efficient system which can detect anomaly action during an online exam such as looking to the side, talking and using the phone. A suspicious movement is only logged when it lasts more than 3 seconds to avoid random model falsity.

3.1. Front camera results

This work can detect if a student is peeking abnormally to the right or to the left.

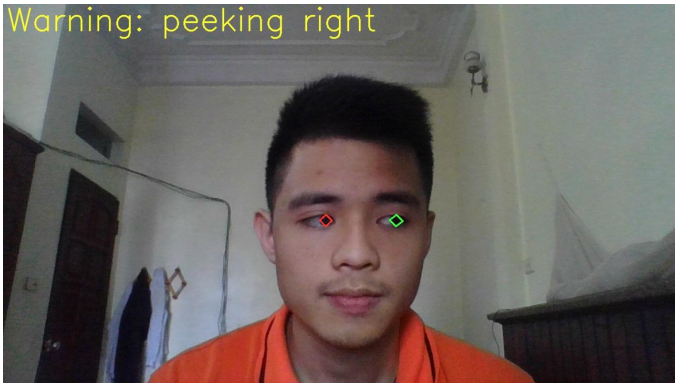


Figure 18. *Eyes tracking result*

Fig. 18 shows the result of the student's eyes tracking. Attempts to peek abnormally and exceed the predefined eye ratio will be flagged with corresponding eyes direction.

Abnormal head-turning of a student to the right, left or down can also be detected with a precise angle degree.

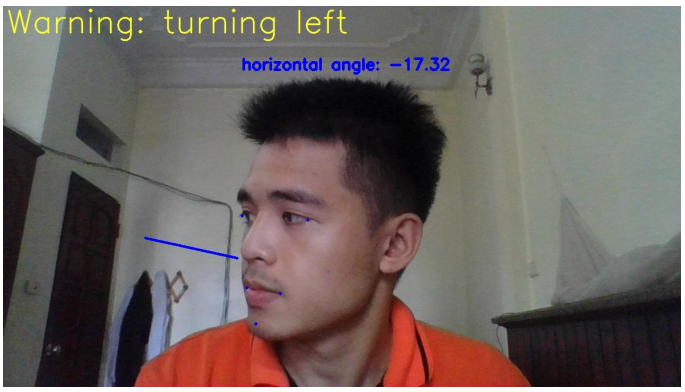


Figure 19. *Head pose estimation*

Fig. 19 shows the head pose estimation result. Using vertical and horizontal angles returned by the PnP method, the model can track which direction a student's head is turning and to what extent.

This work can also detect if a student is talking to others or hiding his face in trying to cheat.

Figure 20. *Mouth tracking and face occlusion detected*

Fig. 20 shows tracking results of student mouth and hands. Flags are raised here because the model detects the increase of distances between lips and covering of hand coordinates in lips coordinate, suggesting the test taker's attempt to talk during the exam.

3.2. Side camera results

After 50 epochs, the pose recognition model gives 93.6 percent accuracy and 0.16 loss on training sets. The below graph depicts the total training process.

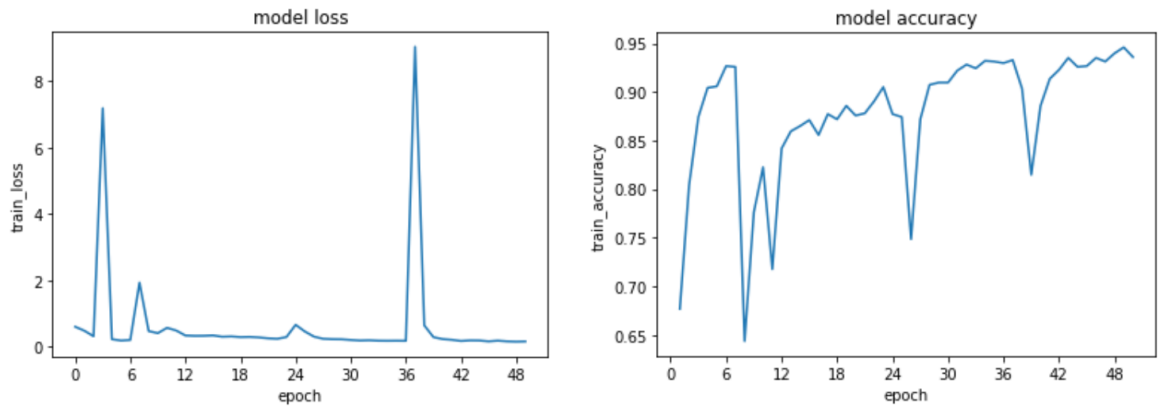


Figure 21 *. Model loss and accuracy on training dataset*

On the test set, the model generalizes with an accuracy of 95.8% in which 1 example is misclassified as cheating and 5 examples are misclassified as not cheating.

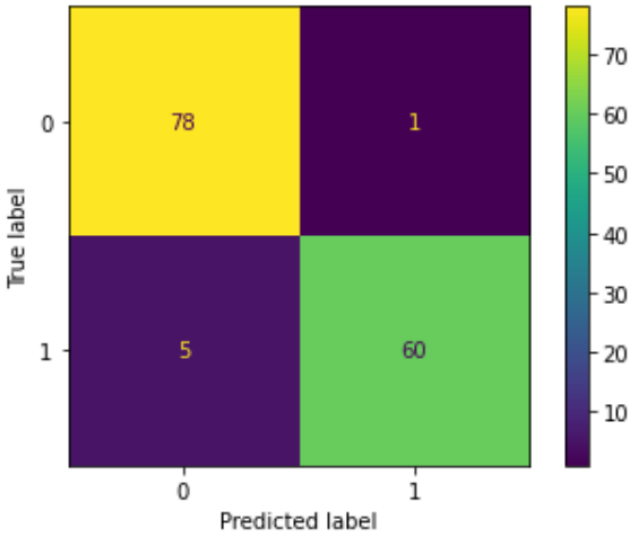


Figure 22. *Confusion matrix to evaluate model accuracy*

The test accuracy gives the confidence to apply in real-life scenarios, below graphs show instances predicted by this work's model.

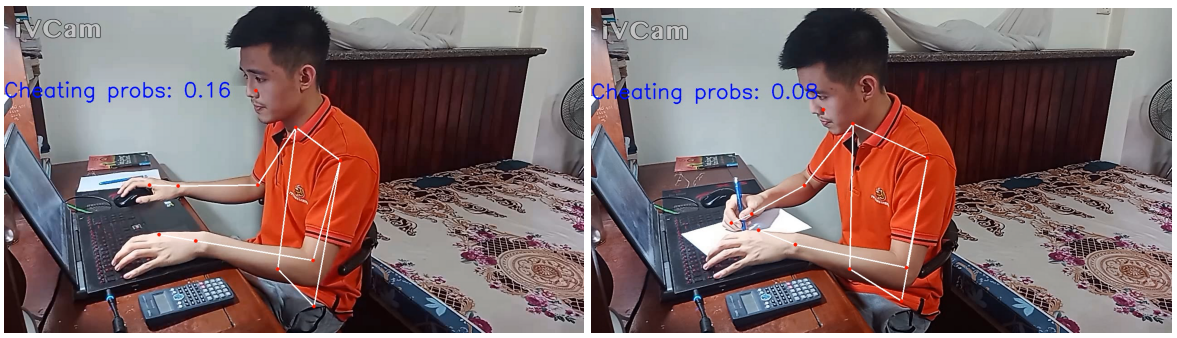


Figure 23. *No suspicious action was detected*

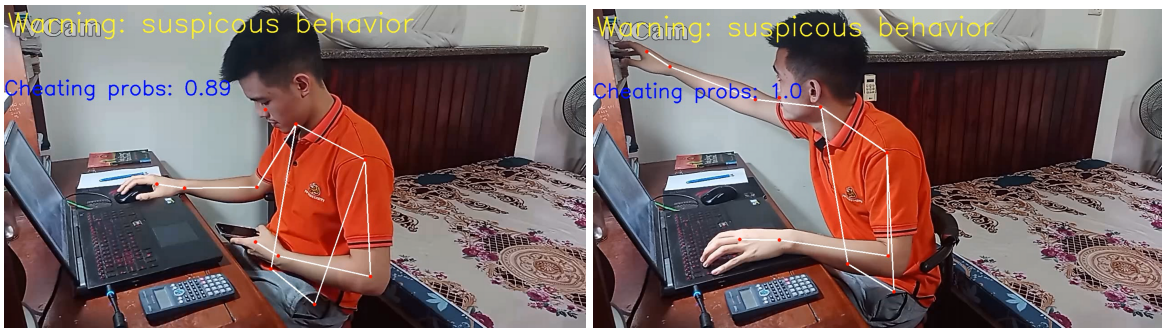


Figure 24. *Suspicious action detected*

Fig. 23 and Fig. 24 show the output of the trained pose recognition model. With suspicious action, the model gives high confidence in cheating behavior. The same happens as the model predicts non-cheating behavior when a student is focused on doing the test.

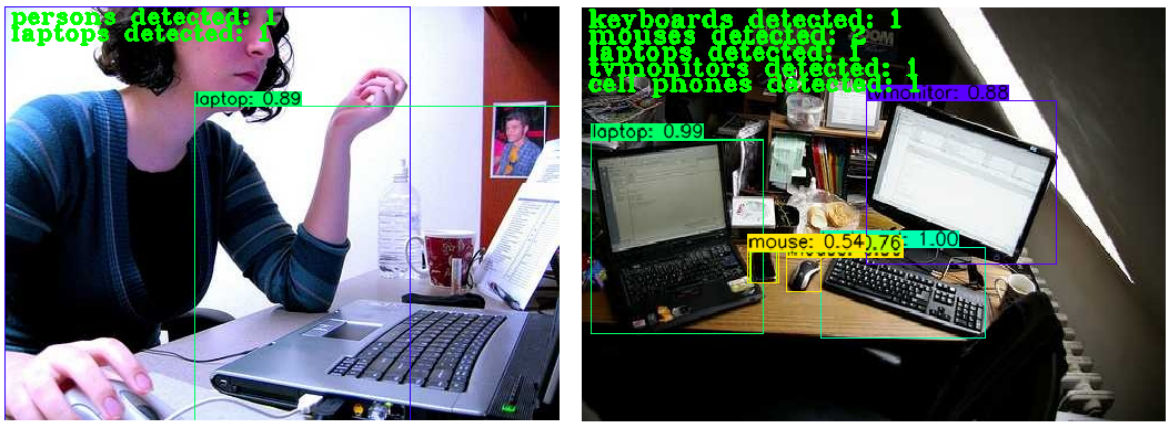


Figure 25. *Results of object detection and person counting*

The object detector used has high reliable accuracy. The detector can identify and count recognizes 8 types of objects for this specific case - 8 classes include: person,

tv-monitor, laptop, mouse, remote, keyboard, cell phone, book. The accuracy of the model is approximately 90 percent with the pre-trained network.

As a student is flagged with any suspicious action, this system automatically logs them back for a 10-second long duration with a corresponding name. Official proctors can then check-in details with log notifications.

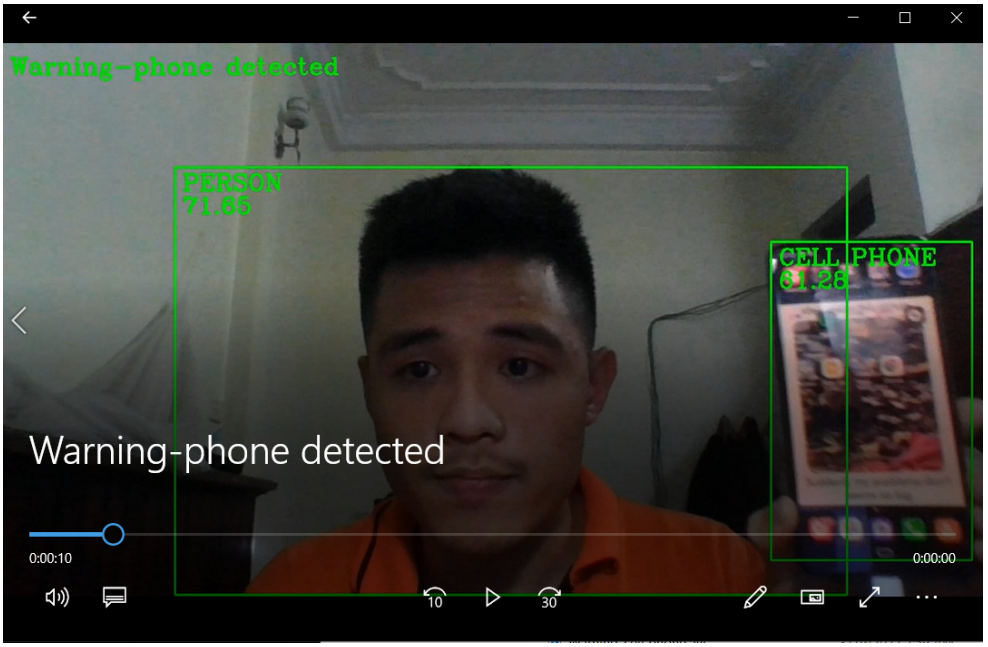


Figure 26. *Video logged by the system*

4. Conclusion and future works

4.1. Conclusion

This study proposes a proctor assistant system that can detect students' anomaly actions through cameras and record them using computer vision. This system shows promising results and can be used to aid teachers and proctors when monitoring the examination. The system saves a lot of time and effort for the organization.

4.2. Future work

Due to the present covid 19 circumstances, there is an urgent need to apply an intelligent system to monitor test-takers during online exams and stop many preventable cheating attempts. Many improvements can be made based on the

proposed model including automatic identification of student identities based on the university's face database; customizing object recognition to the specific requirements of the various exams.

The results continue to be improved as more new real-life situations are added. Machine learning architecture can also be tuned, and the system is aimed at ease of installation on many different hardware platforms and operating systems.

References

1. University of Melbourne: Zoom-supervised exams. https://students.unimelb.edu.au/your-course/manage-your-course/exams-assessments-and-results/exams/how-do-i-take-my-exam/formats/Zoom-supervised-exams. Accessed: 2022/03/01.

2. Government College University Faisalabad: Instructions for Online Exam. https://gcuf.edu.pk/notification-single?news=323. Access: 2022/03/01.

3. FPT University: Exam software announcement and preparation for the exam on EOS software (in Vietnamese)

   https://docs.google.com/document/d/1SZgQL5WQRL7VznXKpb6t369AqD8vYqlW/edit. Accessed: 2022/03/01

4. Søgaard, T.M.: Mitigation of cheating threats in digital BYOD exams. Master's thesis, NTNU (2016). https://dx.doi.org/11250/2310735

5. Kanchan, R.: 7 Online Proctoring Technologies That Guarantee High Test Integrity. https://blog.mettl.com/prevent-cheating-in-online-exams/ Accessed: 2022/03/01.

6. Bedford, D.W., Gregg, J.R., Clinton, M.S.: Preventing online cheating with technology: A pilot study of remote proctor and an update of its use. Journal of Higher Education Theory and Practice 11, no. 2, 41-59 (2011)

7. Motwani, S., Nagpal, C., Motwani, M., Nagdev, N., Yeole, A.: AI-Based Proctoring System for Online Tests. In: Proceedings of the 4th International Conference on Advances in Science & Technology (ICAST2021), Available at SSRN: https://ssrn.com/abstract=3866446 or http://dx.doi.org/10.2139/ssrn.3866446 (2021)

8. Atoum, Y., Chen, L., Liu, A.X., Hsu, S.D., Liu, X.: Automated online exam proctoring. IEEE Transactions on Multimedia 19, no. 7, 1609-1624 (2017)

9. Turani, A.A., Alkhateeb, J.H., Alsewari, A.A.: Students Online Exam Proctoring: A Case Study Using 360 Degree Security Cameras. In: Proceedings of the Emerging Technology in Computing, Communication and Electronics (ETCCE), pp. 1-5, doi: 10.1109/ETCCE51779.2020.9350872 (2020)

10. Kartynnik, Y., Ablavatski, A., Grishchenko, I., Grundmann, M.: Real-time facial surface geometry from monocular video on mobile GPUs. arXiv:1907.06724 (2019)

11. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

12. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C-L., Yong, M.G., Chang, W-T., Hua, W., Georg, M., Grundmann, M.: Mediapipe: A framework for building perception pipelines. arXiv:1906.08172 (2019)

13. Ablavatski, A., Vakunov, A., Grishchenko, I., Raveendran, K., Zhdanovich, M.: Real-time Pupil Tracking from Monocular Video for Digital Puppetry. arXiv:2006.11341 (2020)

14. Rocca, F., Matei M., Bernard, G.: Head pose estimation by perspective-n-point solution based on 2d markerless face tracking. In: Proceedings of the International Conference on Intelligent Technologies for Interactive Entertainment. Springer, Cham (2014)

15. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C-L., Crundmann, M.: Mediapipe hands: On-device real-time hand tracking. arXiv:2006.10214 (2020)

16. Bochkovskiy, W., Wang, C-Y., Liao, H-Y.M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934v1 (2020)

17. Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., Grundmann, M.: Blazepose: On-device real-time body pose tracking. arXiv:2006.10204 (2020)

Appendix

Demo code

Below are some of the code demonstrating the approach presented in this thesis.

```
#Return landmarks data from Mediapipe
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
    image.flags.writeable = False
    results = model.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results

#Return scaled landmark because the Mediapipe range is (0,1)
def get_scaled_landmarks(landmarks, dimension):
    landmarks_2d = []
    landmarks_3d = []
    if dimension == '2d':
        for landmark in landmarks:
            x, y = int(landmark.x*1280), int(landmark.y*720)
            landmarks_2d.append([x, y])
        return landmarks_2d
    if dimension == 'both':
        for landmark in landmarks:
            x, y = int(landmark.x*1280), int(landmark.y*720)
            landmarks_2d.append([x, y])
            landmarks_3d.append([x, y, landmark.z])
        return landmarks_2d, landmarks_3d

#Get current eyes position
def get_eyes_movement(eyes_landmarks):
    eyes_landmarks = get_scaled_landmarks(eyes_landmarks, '2d')
    right_most, right_iris, right_inner, left_inner, left_iris, left_most = eyes_landmarks
    try:
        right_ratio = abs(right_iris[0]-right_most[0]) / abs(right_most[0]-right_inner[0])
        left_ratio = abs(left_iris[0]-left_inner[0]) / abs(left_inner[0]-left_most[0])
    except:
        return 0
    #look right
    if right_ratio < 0.35 and left_ratio < 0.35:
        return "peeking right"
    #look left
    elif right_ratio > 0.65 and left_ratio > 0.65:
        return "peeking left"
    else:
        return ""

#Get current head movement
def get_head_movement(image, face_keypoints):
    face_2d, face_3d = get_scaled_landmarks(face_keypoints, 'both')
    nose_2d = face_2d[0]
    face_2d = np.array(face_2d, dtype=np.float64)[1:]
    face_3d = np.array(face_3d, dtype=np.float64)[1:]
    focal_length = 1*iw
    cam_matrix = np.array([[focal_length, 0, ih/2],
```

```
                    [0, focal_length, iw/2],
                    [0, 0, 1]])
    dist_matrix = np.zeros((4,1), dtype=np.float64)
    _, rot_vec, _ = cv2.solvePnP(face_3d, face_2d, cam_matrix, dist_matrix)
    rmat, _ = cv2.Rodrigues(rot_vec)
    angles, _, _, _, _, _ = cv2.RQDecomp3x3(rmat)
    x = angles[0] * 360
    y = angles[1] * 360
    if y < -15:
        text = "turning left"
    elif y > 15:
        text = "turning right"
    elif x <-10:
        text = "turning down"
    else:
        text = ""
    return text

#Check if student is talking
def get_mouth_movement(upper_mouth, bottom_mouth):
    distance = abs(int(upper_mouth.y*ih) - int(bottom_mouth.y*ih))
    if distance > 15:
        return "talking"
    else:
        return ""

#Check if student is using hand to hide face
def get_hand_movement(face_hands, hand_limit):
    x_min, x_max, y_min, y_max = hand_limit
    face_hands = get_scaled_landmarks(face_hands, '2d')
    warning = ""
    for lm in face_hands:
        if lm[0] > x_min and lm[0] < x_max and lm[1] > y_min and lm[1] < y_max:
            warning = "face occlusion"
            break
    return warning

#Main of first camera
cap = cv2.VideoCapture(0)
eyes_movements, head_movements, mouth_movements, hand_movements= [], [], [], []
warnings = [""]
with mp_hands.Hands(model_complexity=0) as hands:
    with mp_face_mesh.FaceMesh(max_num_faces=1, refine_landmarks=True) as face_mesh:
        while cap.isOpened():
            success, image = cap.read()
            if not success:
                print('Ignore empty camera frame!')
                break
```

```
        image = cv2.flip(image, 1)
        image, results = mediapipe_detection(image, face_mesh)
        image, hand_results = mediapipe_detection(image, hands)
        face_hands = []
        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                lmks = face_landmarks.landmark
                eyes_landmarks = [lmks[263], lmks[473], lmks[362], lmks[133], lmks[468], lmks[33]]
                eyes_movement = get_eyes_movement(eyes_landmarks)
                eyes_movements.append(eyes_movement)
                eyes_movements = eyes_movements[-60:]
                face_keypoints = [lmks[1], lmks[33], lmks[263], lmks[61], lmks[291], lmks[199]]
                head_movement = get_head_movement(image, face_keypoints)
                head_movements.append(head_movement)
                head_movements = head_movements[-60:]
                mouth_movement = get_mouth_movement(lmks[13], lmks[14])
                mouth_movements.append(mouth_movement)
                mouth_movements = mouth_movements[-60:]
                if len(eyes_movements) == 60:
                    warn_eyes = warning_3s(eyes_movements)
                    warn_head = warning_3s(head_movements)
                    warn_mouth = warning_3s(mouth_movements)
                    warnings.extend([warn_eyes, warn_head, warn_mouth])
                draw_face_landmarks(image, face_landmarks)
                face_occlusion_points = [lmks[13], lmks[8]]
                if hand_results.multi_hand_landmarks:
                    for hand_landmarks in hand_results.multi_hand_landmarks:
                        hand_lmks = hand_landmarks.landmark
                        hand_limit = get_limit_hand_coordinate(hand_lmks)
                        hand_movement = get_hand_movement(face_occlusion_points, hand_limit)
                        hand_movements.append(hand_movement)
                        hand_movements = hand_movements[-60:]
                        if len(hand_movements) == 60:
                            warn_hand = warning_3s(hand_movements)
                            warnings.append(warn_hand)
                        draw_hand_landmarks(image, hand_landmarks)
                else:
                    hand_movements.append("")
                    hand_movements = hand_movements[-60:]
        warning_info, warnings = warning_display(warnings)
        cv2.putText(image, warning_info, (7, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (45, 255,
255), 2, cv2.LINE_AA)
        cv2.imshow(" Output feed", image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            cv2.imwrite("hand.jpg", image)
            break

#Calculate angles from 3 landmarks
```

```
def get_joint_angle(a, b, c):
    angle = np.abs(np.arctan2(c.y-b.y, c.x-b.x) - np.arctan2(a.y-b.y, a.x-b.x))
    if angle > np.pi:
        angle = 2*np.pi-angle
    return angle

#Get angles represent student elbow and shoulder
def get_all_angles(landmarks):
    nose = landmarks[mp_pose.PoseLandmark.NOSE.value]
    right_shoulder = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value]
    right_elbow = landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value]
    right_wrist = landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value]
    left_shoulder = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
    left_elbow = landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
    left_wrist = landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
    right_elbow_angle = get_joint_angle(right_shoulder, right_elbow, right_wrist)
    righ_shoulders_angle = get_joint_angle(right_elbow, right_shoulder, left_shoulder)
    left_elbow_angle = get_joint_angle(left_shoulder, left_elbow, left_wrist)
    left_shoulders_angle = get_joint_angle(left_elbow, left_shoulder, right_shoulder)
    angles = [right_elbow_angle, righ_shoulders_angle, left_elbow_angle, left_shoulders_angle]
    return angles

#Extract all 204 landmarks data
def get_frame_landmarks(results):
    size_landmarks = np.array([[res.x, res.y, res.z, res.visibility] for res in
results.pose_landmarks.landmark[:23]]).flatten() if results.pose_landmarks else np.zeros(4*25)
    world_landmarks =  np.array([[res.x, res.y, res.z, res.visibility] for res in
results.pose_world_landmarks.landmark[:23]]).flatten() if results.pose_world_landmarks else
np.zeros(4*25)
    angles = np.array(get_all_angles(results.pose_landmarks.landmark)) if results.pose_landmarks else
np.zeros(4)
    landmarks = np.concatenate([size_landmarks, world_landmarks, angles])
    return landmarks

#Main of second camera
input_sequence = []
cap = cv2.VideoCapture(1)
frame_num = 0
with mp_pose.Pose() as pose:
    while cap.isOpened():
        # Read feed
        ret, frame = cap.read()
        if not ret:
            print("Can't get frame!")
            break
        frame_num += 1
        # Make detections
        image, results = mediapipe_detection(frame, pose)
```

```
        #Draw landmarks
        if results.pose_landmarks:
            draw_landmarks(image, results)
        frame_landmarks = get_frame_landmarks(results)
        input_sequence.append(frame_landmarks)
        input_sequence = input_sequence[-30:]
        if len(input_sequence) == 30:
            res = model.predict(np.expand_dims(input_sequence, axis=0))[0]
            cheating_prob = round(res[1], 2)
            cv2.putText(image, "Cheating probs: "+str(cheating_prob), (0, 200),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, ((255, 0, 0)), 2, cv2.LINE_AA)
            if cheating_prob > 0.8:
                cv2.putText(image, "Warning: suspicious behavior", (7, 60),
cv2.FONT_HERSHEY_SIMPLEX, 2, (45, 255, 255), 2, cv2.LINE_AA)
        #Show image feed
        cv2.imshow('OpenCV Feed', image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```