American Academic Scientific Research Journal for Engineering, Technology, and Sciences
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
http://asrjetsjournal.org/

An Anti-Cheating System for Online Interviews and Exams

Azmi Can Özgen[a]*, Mahiye Uluyağmur Öztürk[b], Umut Bayraktar[c], Selim Aksoy[d]

[a,b,c]*Huawei Turkey R&D Center, Istanbul, Turkey*
[d]*Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey*
[a]*Email: azmican73@gmail.com*
[b]*Email: mahiye.uluyagmur.ozturk@huawei.com*
[c]*Email: umut.bayraktar@huawei.com*

Abstract

Remote examination and job interviews have gained popularity and become indispensable because of both pandemics and the advantage of remote working circumstances. Most businesses and educational organizations use these platforms for recruitment as well as online exams. However, one of the critical problems of the remote examination systems is conducting the exams in a reliable environment. In this work, we present a cheating analysis pipeline for online interviews and exams. The system only requires a video of the candidate, which is recorded during the exam by using a webcam without a need for any extra tool. Then cheating detection pipeline is employed to detect the presence of another person, electronic device usage, and candidate absence status. The pipeline consists of face detection, face recognition, object detection, and face tracking algorithms. To evaluate the performance of the pipeline we collected a private video dataset. The video dataset includes both cheating activities and clean videos. Ultimately, our pipeline presents an efficient and fast guideline for detecting and analyzing cheating actions in an online interview and exam video.

Keywords: Cheating detection; Face detection; Object detection; Face tracking; Video processing.

1. Introduction

Nowadays, online job interviews are becoming increasingly popular. One rationale for this type of hiring is pandemic preparation, but the ability to spend time and energy effectively is also a key motivator for both individuals and employers. Instead of conducting interviews in actual offices, candidates may now do interviews online at any time and from any location globally.

---

* Corresponding author.

As a consequence, online interviews make hiring faster and easier. Exams are popular instruments used to assess people's understanding of specific areas. However, cheating is a critical problem and must be eliminated to ensure academic honesty [1]. As a result, proctoring online exams in a dependable setting is critical for the exams to be authentic. According to King and Case [2], over 74% of students believe they can easily cheat in online exams. When exams are administered in the traditional manner, applicants are to be overseen by a human proctor, which is often utilized to provide credible exams. However, proctoring is a tedious and time-consuming task. It simply entails keeping an eye on an individual or a group of people throughout an exam and preventing pre-planned cheating behaviors. Even after the pandemic has passed, automating the proctoring procedure at universities or colleges can allow highly educated staff to work more efficiently. Automated proctoring solutions, in essence, are valuable additions to online interview and exam systems. To validate the integrity of the interview, online interview systems that capture candidate videos often use computer vision and machine learning algorithms [3,4,5]. Face verification is a critical component of these systems. Furthermore, object detection is required to track any other irregularities that may arise throughout the session. It aids in detecting the appearance of other people or the prohibited use of an electronic device.

It may be argued that these cheating activities do not have to take place in front of cameras and may never be discovered. Although this is a valid point, our system is designed to process only video records. We believe it would be helpful to use it as part of a larger interview system in order to dissuade candidates from cheating.

We created an online interview anti-cheating solution that is powered by cutting-edge deep learning detection models. A system like this could be utilized in an online exam where the candidate answers questions that have already been prepared without the need for a human proctor. For this aim, the candidate's frontal view, as shown in Figure 1, should be captured for subsequent examination.

Our goal is to detect the following occurrences that may occur during the exam: (1) the presence of another person, (2) the use of an electronic device, and (3) the absence status. Figure 2 depicts an exam environment with a candidate and various disallowed interactions and actions that they must avoid for a genuine interview or exam.

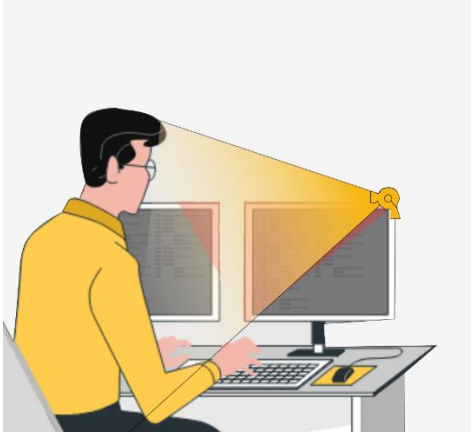

Figure 1: Cheating detection system.

Our contributions are summarized below:

- We have created an efficient cheating detection pipeline.
- We have deployed a combination of image/video processing techniques particularly designed to detect cheating activities.
- We have presented cheating evaluation components such as another person's appearance, electronic device usage, and candidate absence status to determine a cheating activity.
- We have employed state-of-the-art deep learning models for face and object detection as cheating detection components.
- We have proposed three metrics to evaluate the performance of the pipeline.

An earlier version of this work was presented in [6]. We included more detailed literature review, dataset evaluation, experiments, performance metrics and graphical analysis as major differences.

Figure 2: An exam environment with forbidden interactions (use of an electronic device and presence of an another person).

2. Related Works

Poutre and his colleagues [7] and Arno and his colleagues [8] provide a comprehensive comparison of online proctoring systems and their features. Examples of such systems include online exam environments that can prevent cheating acts with preemptive procedures, allowing them to reduce the number of cheating attempts without the use of a proctor [9,10,11]. Another form of this technique combines preemptive systems with human proctors [12,13].

There are automated interview systems without AI-based anti-cheating services [14,15]. These methods were created to evaluate candidates for an online coding platform. Although some form of plagiarism detection is provided [14,15], there is no detection for the presence of several people or the use of electronic devices.

Some applications, such as [16], use simply a user verification system with a trajectory analysis to verify that the person is the actual person who has been granted access to the system. Other alternatives, such as [17], combine automated detection systems with human proctors, as well as well-defined workflows. Furthermore, Asep and Bandung [5] suggest a user verification system for online exam proctoring.

Multi-modal techniques combine different cheater analysis methodologies in visual and audial features. Reference [18] employs these characteristics in conjunction with psychological factors to determine a candidate's hirability. Some multi-modal techniques, such as [3,4], present detailed pipelines with features such as voice detection, gaze estimation, and head-pose estimation. This type of broad pipeline, on the other hand, often necessitates greater computer capacity.

Online proctoring technologies powered by artificial intelligence have a significant impact on exam administration in a trustworthy and safe environment [3,19]. Our system has a similar structure to the study [3], including detection of another person and electrical devices. In addition, in [3], an eye tracker was utilized to determine where the candidate was looking during the exam. In [20], a dual vision camera is utilized to capture additional information from a person's face. However, providing an eye tracker or dual vision camera to all candidates in a typical exam setting is often not possible.

3. Methodology

Our system aims to detect predefined cheating activities in a fully automated way by processing only the recorded video of the test-taker or -as we call- *Candidate*. The system does not require any additional hardware except the webcam or any available camera. Moreover, there is no authentication phase where the candidate uploads some visual and textual information in advance. We only need the candidate to show themselves clearly in the first 20 seconds of the video to let us register the candidate's face for further process. All these features make our system quite fast and efficient as we delve into detail in the following sections.

The following subjects are covered in the remainder of this section: Dataset, Cheating detection pipeline, Video pre-processing, Candidate's face detection, Face recognition, Object detection, Face tracking, and Result analysis.

3.1. Dataset

Because there is no publicly available dataset for cheating detection, we built and created our own data collection and labeling strategy. We used our online exam platform[1], which measures participants' talents for employers with hundreds of questions drawn from a question pool. The pool contains both multiple-choice and free coding problems. We asked the participants to engage in some deliberate cheating. These behaviors include having someone other than the candidate in the exam area, the candidate leaving the exam environment, visibly utilizing a cell phone or laptop, or any combination of these occurrences throughout the video. Participants took part in the trials from various locations such as rooms, schools, or workplaces; they employed various camera

[1] www.talent-interview.com/

angles and settings; they were in varying lighting conditions. These variations made detecting cheating events quite difficult; for example, mobile phones were barely visible because participants held them right next to their ears in a speaking position without clearly showing the phone to the camera, or the candidate left the exam partially without showing their face by showing a part of their body.

From Huawei Turkey R&D Center, 22 participants with 37 videos joined our experiments. As previously stated, the majority of them were requested to purposefully engage in a cheating event. The videos range in length from 10 to 25 minutes. Two annotators observed and labeled cheating intervals in seconds (start second, end second). Each incident was labeled individually. For example, if *Absence* and *Another person* events occurred simultaneously, they are both labeled as two events. To be labeled for the *Another person* occurrence, the other person's face or body must be visible. Similarly, for the *Absence* event, the candidate's face and body should be absent from the scene. Using a mobile phone or laptop without being seen in the camera view was not considered a cheating event for the *Device* event; nonetheless, even if a little part of the device was visible, it was annotated. Figure 3 depicts a pie chart of our dataset with the counts of cheating incidents.

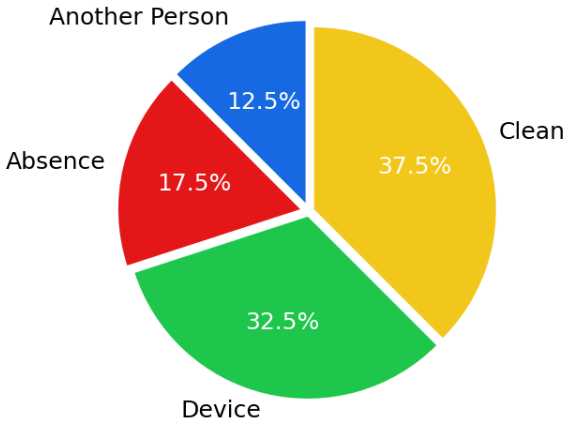

Figure 3: Cheating event distribution of our dataset with 37 videos.

As previously stated, even slight variations in camera angles, lighting, and visibility rates cause significant variances in events. Other variations evolved as a result of the occurrence of the cheating events in various ways. For example, a candidate may exit the exam environment and another individual may become visible for a limited period of time. Alternatively, a candidate could demonstrate the use of an electronic device just by displaying a portion of its cover side. These contentious cases spark debate during both the labeling and testing phases

3.2. Cheating detection pipeline

We make several assumptions for an ideal online interview system in order to define cheating instances that we want to detect. To begin, we suppose that there is a person whose face appears throughout the video and who is referred to as *Candidate*. A genuine online interview video, in which no instances of cheating actions are observed, is characterized as:

- There should not be any people other than one Candidate.
- Using a mobile phone or an additional computer is strictly prohibited. However, we can only detect them if they are visible in the video.
- Leaving the exam environment for more than 5% of the exam duration is forbidden.

Violating at least one of the situations above will result in cheating actions as *Another person*, *Device*, and *Absence* respectively, and the overall label of the video will be *Suspicious*. If all three actions are clean, then the video is labeled as *Clean*. Due to the subjective character of cheating occurrences, we favor the descriptor *Suspicious* over *Cheating*.

Our anti-cheating pipeline attempts to decide on each of these three events and it consists of five major elements: *Video pre-processing*, *Candidate's face detection*, *Face recognition*, *Object detection* and *Result analysis*. Details of the pipeline can be seen in Figure 4.

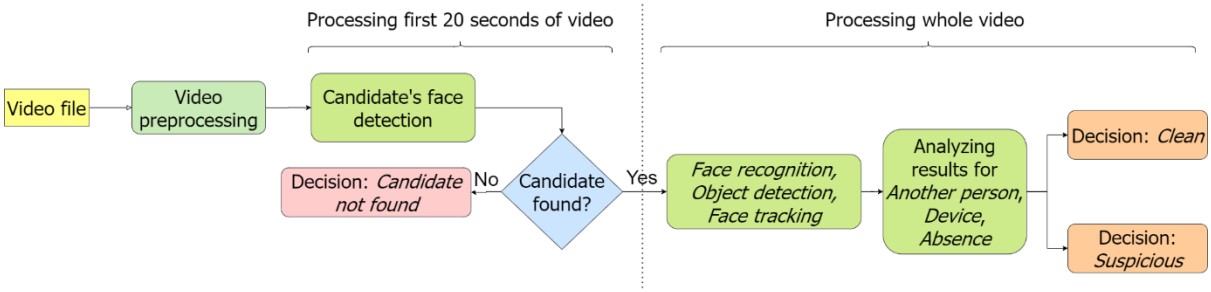

Figure 4: Proposed cheating detection pipeline.

1) The video file is preprocessed for the standard codec, constant fps, and resolution values. 2) The candidate's face is identified and registered in the first 20 seconds of the video for further analysis. If the candidate's face cannot be found, then the video will not be processed further. 3) Face detection, face tracking, and object detection modules traverse the entire video, processing each frame. Each frame's outcome is provided in a table. 4) Each cheating event field, *Another person*, *Device*, and *Absence*, is investigated in the table. 5) Each field is labeled as either *Clean* or *Suspicious*. 6) If at least one field is labeled as *Suspicious* then the final output is also *Suspicious*; otherwise, it is *Clean*.

3.3. Video pre-processing

Data pre-processing is an essential step for these kinds of applications. Therefore, videos must be pre-processed for the succeeding steps to operate more efficiently. In this work, all videos are converted to pre-defined format with fixed resolution.

Other than resolution, FPS (frame per second) rate is an important parameter at this step. We set our videos at a constant 3 FPS rate. Generally, higher FPS rates mean higher accuracy. However, for extracting important cheating relations between subsequent frames, this FPS rate was found to be sufficient in our experiments. In addition, 400 pixel width was employed while maintaining their original aspect ratios.

3.4. Candidate's face detection

One major feature of our system is the authorization of the candidates by processing the first 20 seconds of the video. This eliminates the identity checking procedure before starting video recording. Therefore, locating and registering the candidate's facial encodings is necessary for this step.

Histogram of Oriented Gradient (HOG) [21] based Support Vector Machine (SVM) is a well-known algorithm for face detection. Besides, Convolutional Neural Networks (CNN) based detectors are very popular. We have employed these two methods at this stage and used pre-trained models from the dlib [22] library for both of them. The candidate's face and face encodings are determined for future analysis. At this point, we have only used the HOG detector. We locate all of the faces in each frame in the first 20 seconds of the video. Head-pose estimation is a supportive technique that is used in face detection applications frequently to detect face angles. We utilized it for examining the face angles. Here, the angles between the camera's horizontal and vertical axes and the face pose axes should be measured. If the measured angles exceed the thresholds, then those faces are not acknowledged as candidate's faces. HOG-based SVM classifier generates $128D$ vectors for all obtained faces for future comparison throughout the video.

3.5. Face recognition

We process each frame of the video to detect the existence of the candidate's face, count the faces and bodies, and detect the appearance of an electronic gadget. At this point, the HOG-based detector is the primary detector. If it is unable to detect any faces, a CNN-based detector is used to look for possible faces. The former is chosen for shorter processing times.

While traversing the frames, we compare the encodings of the candidate's face to each located face in a frame. As previously stated, registered frames from the first 20 seconds of the video are gathered and are referred to as F_s, where s denotes the frame index in the first 20 seconds. Traversed frames are called F_t, where t represents the frame indices of the entire video. Face distance is calculated for each identified face in any frame, F_t, as

$$\min\left(||D_1||_F, ||D_2||_F, \dots, ||D_s||_F, \dots, ||D_n||_F\right) \tag{1}$$

where $\boldsymbol{n = 60}$ is the number of frames collected from the first 20 seconds of **3** FPS video, and $||\boldsymbol{D_s}||_{\boldsymbol{F}}$ is the Euclidean distance between $\boldsymbol{F_s}$ and $\boldsymbol{F_t}$ as

$$||D_s||_F = ||F_s - F_t||_F \tag{2}$$

which can also be written as

$$||D_s||_F = \sqrt{\sum_{i=1}^{128} ||F_s - F_t||_F}\,. \tag{3}$$

The distance to the nearest face among all registered faces is considered as the face distance, as in (1). We also

used *partial face matching*, a basic yet effective technique for partially viewed faces. When we compute the distance between $\boldsymbol{F_s}$ and $\boldsymbol{F_t}$ in (2), we make the assumption that values lower than $\boldsymbol{\epsilon = 10^{-2}}$ in $\boldsymbol{F_t}$ are coming from unseen or imperceptible features of the detected face. By equalizing $\boldsymbol{F_t}$ and $\boldsymbol{F_s}$ at the same indices in the $\boldsymbol{128D}$ face encodings vector, we eliminate the distances resulting from these small values. In other words, small values in $\boldsymbol{F_t}$ are disregarded while computing the face distance. This assumption reduces many false alarms by allowing lower face distances when a candidate is present but only partially visible.

A constant thresholding approach is used to calculate outlier face distances. Frames with facial distances that exceed the $\boldsymbol{0.65}$ threshold in our experiments are nominated as frames with another person. Following that, we link those frames to each other based on their proximity in the time sequence to determine how many times another person appears in the video. If those frames are connected, each sequence counts as $\boldsymbol{1}$ appearance of another person.

3.6. Object detection

Selected object classes to be detected are *person*, *mobile phone* and *laptop* from the COCO2017 dataset [23]. Mobilenet-SSD model [24] was trained with these three classes to run on each traversal frame $\boldsymbol{F_t}$.

Unlike face detection, confidence values are used from the Mobilenet-SSD model to determine cheating actions rather than distance values. Body detection threshold ($\boldsymbol{0.65}$) and device detection threshold ($\boldsymbol{0.30}$) were used in our experiments. These are optimum threshold values based on our experiments in order to achieve a balance of real detections and false alarms.

Body counts and confidence values are collected from each frame for body detection. Nominated frames are selected by the following conditions: (1) The frames with a body count of $\boldsymbol{1}$ but no candidate's face is found. (2) Body count is more than $\boldsymbol{1}$ but a candidate's face is found. Nominated frames are united as one event based on their distances in the sequence to decide how many times multiple bodies appeared in the video. Throughout the video, if there has been at least $\boldsymbol{1}$ event, the decision for *Another person* is positive.

Modified Mobilenet-SSD that has three classes produces outputs for each of them (body, mobile phone, and laptop). As a result, confidences are thresholded, as in body detection. The frames that exceed the threshold are connected to determine the count of a device that appears. If the count is at least $\boldsymbol{1}$, the conclusion is that existence of a device is positive.

3.7. Face tracking

We also used a face tracking algorithm, the correlation tracker [25], for better face comparison. The method is similar to that of face detection technique and is supplied by the dlib library [22].

Face detection algorithm might fail in frames where the interested face region is not visible clearly. In these situations face tracking can support to find those invisible face regions. These situations frequently arise when the candidate's face is temporarily blocked by an object or when the face is exposed from an unusual angle. In

this scenario, the face tracking result is prioritized. Face tracking is deactivated in circumstances where the candidate's face is blocked or when they depart the environment, resulting in a significant decline in face tracking confidence.

The other critical scenario arises when the faces found by face tracking and face detection do not overlap. If their relative distance (relative to the length of the frame diagonal) is too large, this frame is taken as *Another person*.

3.8. Result analysis

Outputs of every single frame are recorded in a table that is evaluated. This table summarizes the components of all features presented above.

The findings of face and body detection are combined in the *Another person* decision. If one of them has a count of at least **1**, the *Another person* event is reported as *Suspicious*, otherwise it is *Clean*. The *Device* event is reported as either *Clean* or *Suspicious* as a direct result of device detection.

We have a **5**% upper limit for determining if *Absence* occurred or not. If the candidate's face is not discovered in a frame with a body count of **0** and the ratio of the count such frames to total number of frames exceeds a **5**% threshold, report of *Absence* is *Suspicious*. Otherwise, *Clean* is used.

Having at least one of these three events as *Suspicious* makes the overall decision for the video *Suspicious* as well; otherwise, the conclusion is *Clean*.

4. Experimental Results

We executed tests on a private dataset that we collected and labeled from online interviews with **22** different participants, totaling **37** various interview videos ranging in duration from **10** minutes to **25** minutes as explained in Section 3.1. The majority of the videos involve deliberate cheating actions such as another person's appearance, electronic device use, or the candidate's absence.

Videos were watched and labeled based on their cheating events. If at least one of the three events (*Another person*, *Device*, and *Absence*) appears to be dominant, the video is labeled as *Suspicious*, and the related cheating event is also labeled as *Suspicious*.

4.1. Performance metrics

As in [3], we have utilized instance-based and segment-based metrics to measure our performance. Additionally, we have measured overall accuracies by video-based labels. We calculate accuracies by two ratios: True Detection Rate (TDR) and False Alarm Rate (FAR). TDR is the ratio of the number of correct predictions to the number of all positive labels which measures how many of the cheating actions are predicted correctly. On the other side, FAR is the ratio of the number of false predictions to the number of all positive predictions which

measures how many predictions are wrong.

4.1.1. Instance-based evaluation

A true cheating instance is a region with black borders and the detection instance is the red region as drawn in Figure 5. For matching regions, we have used IOU (Intersection over Union) ratio. So, for a cheating instance if the detection instance with the highest IOU of all detection instances is above $\mathbf{0.1}$, then it counts as a true detection. Similarly, for a detection instance if the true cheating instance with the highest IOU of all true cheating instances is below 0.1, then it counts as a false alarm. In Figure 5, TDR is $\mathbf{0.83}$ whereas FAR $\mathbf{0.16}$ by instance metric.

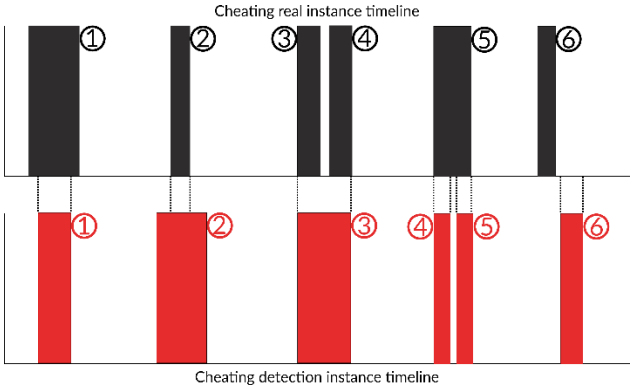

Figure 5: Sample true cheating (black) and detection (red) instances on separate timelines.

Instances 1 and 2 in both timeline show the matching cases even though the detection instances do not exactly overlap with real instances. Cheating instance 6 is a missed and detection instance 6 is a false alarm case. Cheating instances 3 and 4 combined, and instance 5 individually show the cases where a careful matching with predictions is required. In these cases, we match the cheating instance with the closest detection instance and check their IOU values for calculating true detections. Similarly, to be able to decide a false alarm case, we match the detection instance with the closest cheating instance. Therefore, same detection instance may be counted as true detection more than once, or there may be multiple false alarms for the same cheating instance. All these cases emphasize that instance based metric is a bit difficult to measure the real performance of our system since it involves a matching complexity.

Table 1 shows the performance of our system by instance-based metric on our private dataset.

Table 1: Instance TDR and FAR values on our dataset.

	Instance TDR	Instance FAR
Another person	0.639	0.000
Device	0.619	0.020
Absence	0.694	0.131

4.1.2. Segment-based evaluation

Frame sequences are segmented according to constant segment lengths with their reference cheating labels and detection results. We have two frame sequences as cheating sequence and detection sequence. Each frame sequence is an array of binary elements where these elements represent the reference cheating labels and the predictions in the cheating sequence and the detection sequence respectively.

For each segment, these two array elements are compared. Array elements correspond to frames. A segment in the cheating sequence is labeled as a cheating segment when the ratio of positive elements to the length of the segment array is above $\mathbf{0.5}$. Similarly, a segment in the detection sequence is labeled as a detection segment if the ratio of positive elements to the segment length is above $\mathbf{0.5}$.

When a segment is labeled as a cheating segment in the cheating sequence and a detection segment in the detection sequence, then we have a true detection. Oppositely, when a segment is labeled as a detection segment in the detection sequence but not labeled as a cheating segment in the cheating sequence, then we have a false alarm.

Tables 2 and 3 show the performance of our system by segment based metric on our private dataset. Segment lengths are **1** and **3** seconds or **3** or **9** frames since experiments were at **3** FPS.

Table 2: Segment (1 sec) TDR and FAR values on our dataset.

	Segment (1 sec) TDR	Segment (1 sec) FAR
Another person	0.510	0.000
Device	0.424	0.027
Absence	0.783	0.110

Table 3: Segment (3 sec) TDR and FAR values on our dataset.

	Segment (3 sec) TDR	Segment (3 sec) FAR
Another person	0.537	0.000
Device	0.447	0.027
Absence	0.961	0.106

4.1.3. Video-based evaluation

Each video sample has one binary label by their cheating events (*Another person*, *Absence*, and *Device*). If there is at least one true cheating instance and at least one detection instance throughout the frame sequence, this is a true detection. Similarly, if there is no true cheating instance and we have at least one cheating detection, it is a false alarm.

Precision, recall and F1 metrics are presented in Table 4. Positive labels are presumed as *Suspicious* whereas negative labels are presumed as *Clean*. Therefore, true-positive and true-negative cases are correctly predicted *Suspicious* and *Clean* ones respectively, whereas false-positive and false-negative cases are the ones incorrectly predicted as *Suspicious* and *Clean* cases respectively.

Among all three metrics presented above, the segment-based metric is a more clear and lucid metric compared to the instance-based one. The latter requires the matching of cheating and detection instances by using a threshold for intersection over union. The former does not require such matching decisions but may be sensitive to the choice of the segment length.

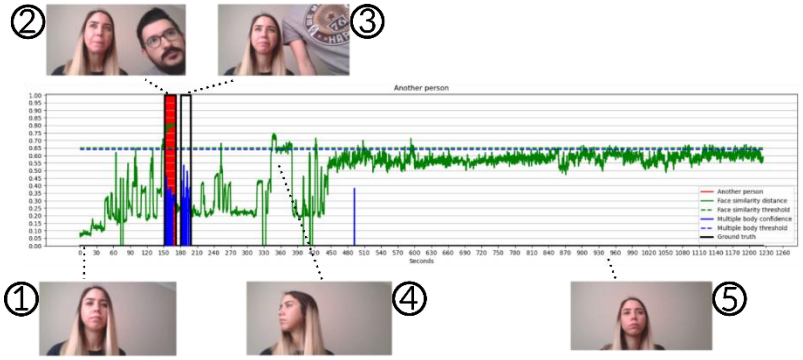

Figure 6: Sequence analysis of Another person instances of a sample video with some corresponding frames.

Black and red colors specify the ground truth and detection instances respectively. Green and blue lines indicate face distance and multiple body confidence values. Dashed lines at 0.65 indicate their threshold values. Frame 1 is the registered candidate face. In frame 2 there is another person and it is detected, whereas in 3 other person's face is not visible and multiple body confidence is below the threshold, and it is a miss. Frame 4 shows a case where the candidate's face angle causes the face distance to increase. However, the face tracking model prevents a false alarm. In the second half of the video, face distance becomes quite large due to the different face position of the candidate as in frame 5 than frame 1. Nevertheless, there is no false alarm at this part.

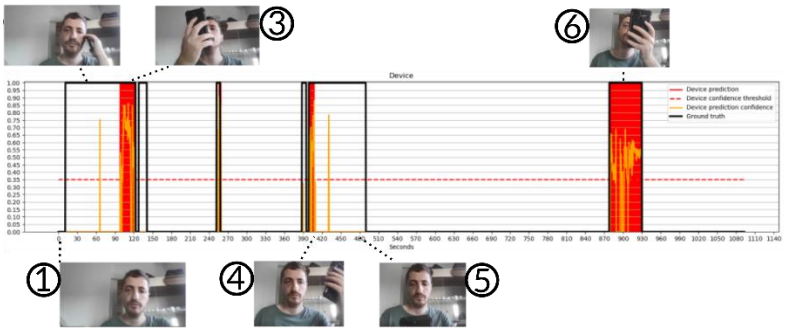

Figure 7: Sequence analysis of *Device* instances of a sample video with some corresponding frames.

Black and red colors specify the ground truth and detection instances respectively. Orange lines are raw confidences of our object detection model and are converted into red regions through sequence analysis. Dashed

lines indicate the threshold value. Frame 1 is the registered candidate face. In frame 2, even though phone call action is obvious, the phone itself is barely visible due to bad angles. In the ending of the same cheating instance, the candidate holds the phone at a better angle as in frame 3, and there we detect it. Frames 4 and 5 show a very similar case as 2 and 3. In frame 6, the phone is apparent once again and the detection is satisfying.

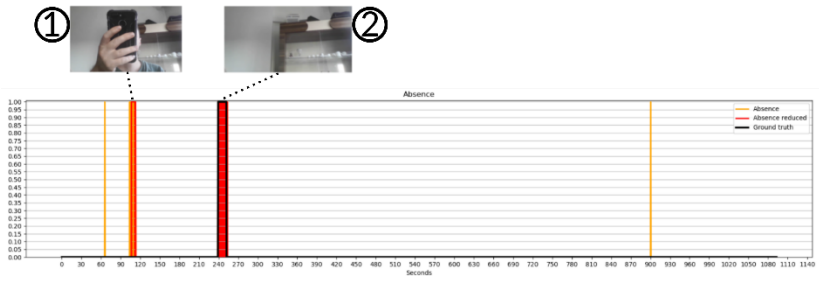

Figure 8: Sequence analysis of *Absence* instances of the same video as in Figure 7 with some corresponding frames on top.

Black and red colors specify the ground truth and detection instances respectively. Orange regions are raw prediction results and eliminated through the sequence analysis part. In frame 1, the device is blocking the face and a false absence alarm occurs. On the other hand, in frame 2, the real absence case is detected correctly.

Although using video-based metric decreases the number of cheating events in the dataset since detecting individual instances are irrelevant, we believe it is a more suitable predilection for measuring system accuracy. Since our system is designed and optimized for detecting cheating activities as quickly as possible without any exhaustive processing, measuring the performance by using a video-based metric is suitable for our system.

In Figures 6, 7 and 8 there are sequence plots from two different sample videos where horizontal axes indicate the time in seconds, and vertical axes define a range between [**0**, **1**] for ground truths and predictions as well as raw predictions of our system. Raw predictions are indicated with orange and they are converted into red regions that are our final prediction instances. There are both success and failure cases shown in Figures 6, 7, and 8. For example, if a candidate's face is shown with an odd angle, or another person's face or body is not clearly seen, then our system might produce wrong predictions.

Our private dataset does not allow us to make a direct comparison to other similar works [3,4]. Nonetheless, individual event-based and overall cheating performances are encouraging.

Relatively lower precision scores in the *Another person* event are caused by higher false-positive rates due to unrecognized candidate faces. There are minor false-negative occurrences in the *Device* event as well, because candidates are mainly trying to disguise their mobile phone usage so that the device is not seen conspicuously.

4.2. Hardware performance

We have used Python[2] language for the implementation of the whole system. All tests were conducted on a workstation with Intel i9 20 CPU cores at 3.60GHz with 64 GB RAM. The operating system was Ubuntu 18.04[3]. Besides, in order to have an easily deployable system to a wide range of machines, we have packaged our environment with Pyinstaller[4] tool, so that all of the required libraries and face/object detection models could be bundled together. The bundle size is around **900** MB. Processing times of all components can be seen in Table 5.

Considering our system can work at constant **3** FPS videos, the overall speed of **4.9** FPS in Table 5 means that the system surpasses real-time speed. In other words, a **1** minute long video is being processed around **37** seconds.

Table 5: Processing times of all components of our system.

	FPS
Video pre-processing	16
Candidate's face detection	498
Face recognition	14
Object detection	47
Result analysis	1.5×10^{-6}
Overall	**4.9**

5. Discussion

In this work, we have focused on designing an efficient pipeline for online interviews and exams with state-of-the-art computer vision and video processing techniques. Since having a fast pipeline is aimed, we had to limit the system inputs and outputs. Inputs are constrained with visual data and audio data is ignored. Furthermore, the system outputs of the cheating analysis are confined just by three cheating events that are *Another person*, *Device* and *Absence*. This makes our system more like a computer vision application.

As mentioned earlier, Figures 6, 7 and 8 show some success and failure cases together. Since we have access only to the frontal view of both candidate and exam environment, if the cheating event is not happening at good visual angles, our system might fail at detecting those cases or produce false alarms. Oppositely, the efficient mixture of face tracking and sequence analysis techniques provides correct predictions. For instance, in Figure 6 frame 4 shows a case where the face tracking module prevents a false alarm by analyzing the continuation in the sequence even though the frontal face of the candidate is not at a desirable angle.

Even though we are only processing visual data, we have employed a combination of classical image and video processing techniques with state-of-the-art deep learning models in an easily deployable and fast algorithmic pipeline. We also claim that the components of the pipeline can be detached and used separately since most of

[2] python.org/downloads
[3] releases.ubuntu.com/18.04
[4] pyinstaller.org

them do not depend on each other. For example, if *Absence* case is not important for an application, that part would be discarded from *Result analysis* phase. Similarly, if one is only interested in the usage of electronic devices in an exam environment, only the object detection module can run, and be analyzed accordingly.

Another advantage of our system is that it does not require any additional hardware or any data source other than a video file. All authentication and processing steps rely on one single video file that is recorded during the exam. On the other hand, an obvious weakness of such an approach is that the inability to detect any cheating action that is out of the scene. Without using auxiliary hardware such as a wearcam as in [3], it is not possible to have all the information of the exam environment. Nonetheless, with this limited environment information and relatively low processing value (**3** FPS), our system promises a reliable assurance at detecting main cheating activities. Moreover, not using those auxiliary hardware makes the system simple, easily applicable, and fast.

Besides, this system can be considered as an application that works collaboratively with human proctors. Since detected cheating event seconds are known, those parts of the video can be trimmed and be checked by the proctor. Such usage is more desirable compared to manually checking the whole video for a possible cheating event.

Further work that can improve the performance of our system would be to employ a voice analysis module as some cheating events may involve only speaking or other cheating events are accompanied by suspicious voice activity. The other improvement is to add a gaze tracking module similar to [26,27] to analyze the eye movements of the candidates. We believe that these two modules can greatly contribute to the accuracy of our system.

6. Conclusion

This paper presents a cheating detection pipeline. The main goal of the system is to provide a reliable and secure exam environment for online interviews and exams. It only requires a small-sized video of the candidate that can be recorded by an integrated webcam. Therefore it is an easily applicable and fast system. The pipeline includes face detection, face recognition, face tracking, and object detection components. We determined three main cheating activities that are *Another person*, *Device*, and *Absence*. We conducted the experiments on a private dataset that consists of **37** videos with real-life cheating activities. As a measurement, we utilized three different metrics that are instance-based, segment-based, and video-based metrics. In the result, we obtained $\mathbf{0.91}$ F1 score by video-based metric. As a further step, we plan to expand our work with voice analysis and gaze estimation features.

References

[1] S. Manoharan, X. Ye, On upholding academic integrity in online examinations, in: 2020 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), IEEE, 2020, pp. 33-37.

[2] D. L. King, C. J. Case, E-cheating: Incidence and trends among college students., Issues in Information Systems 15 (1) (2014).

[3] Y. Atoum, L. Chen, A. X. Liu, S. D. Hsu, X. Liu, Automated online exam proctoring, IEEE Transactions on Multimedia 19 (7) (2017) 1609-1624.

[4] S. Prathish, K. Bijlani, et al., An intelligent system for online exam monitoring, in: 2016 International Conference on Information Science (ICIS), IEEE, 2016, pp. 138-143.

[5] H. S. Asep, Y. Bandung, A design of continuous user verification for online exam proctoring on m-learning, in: 2019 International Conference on Electrical Engineering and Informatics (ICEEI), IEEE, 2019, pp. 284-289.

[6] A. C. Ozgen, M. U. Öztürk, O. Torun, J. Yang, M. Z. Alparslan, Cheating detection pipeline for online interviews, in: 2021 29th Signal Processing and Communications Applications Conference (SIU), IEEE, 2021, pp. 1-4.

[7] B. Poutre, D. Hedlund, W. Nau, Combining testing software, online proctoring and lockdown browsers to assure a secure assessment environment for students in hybrid or online programs (poster 13), Creighton University, Office of Academic Excellence and Assessment (2015).

[8] S. Arno, A. Galassi, M. Tommasi, A. Saggino, P. Vittorini, State-ofthe-art of commercial proctoring systems and their use in academic online exams, International Journal of Distance Education Technologies (IJDET) 19 (2) (2021) 41-60.

[9] G. Cluskey Jr, C. R. Ehlen, M. H. Raiborn, Thwarting online exam cheating without proctor supervision, Journal of Academic and Business Ethics 4 (1) (2011) 1-7.

[10] A. Wahid, Y. Sengoku, M. Mambo, Toward constructing a secure online examination system, in: Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, 2015, pp. 1-8.

[11] J. Opgen-Rhein, B. Küppers, U. Schroeder, An application to discover cheating in digital exams, in: Proceedings of the 18th Koli Calling International Conference on Computing Education Research, 2018, pp. 1-5.

[12] P. Guo, et al., The research and application of online examination and monitoring system, in: 2008 IEEE International Symposium on IT in Medicine and Education, IEEE, 2008, pp. 497-502.

[13] I. Y. Jung, H. Y. Yeom, Enhanced security for online exams using group cryptography, IEEE transactions on Education 52 (3) (2009) 340-349.

[14] G. Anielak, G. Jakacki, S. Lasota, Incremental test case generation using bounded model checking: an application to automatic rating, International Journal on Software Tools for Technology Transfer 17 (3) (2015) 339-349.

[15] S. Vamsi, V. Balamurali, K. S. Teja, P. Mallela, Classifying difficulty levels of programming questions on hackerrank, in: Advances in Decision Sciences, Image Processing, Security and Computer Vision, Springer, 2020, pp. 301-308.

[16] H.-K. Pao, J. Fadlil, H.-Y. Lin, K.-T. Chen, Trajectory analysis for user verification and recognition, Knowledge-Based Systems 34 (2012) 81-90.

[17] X. Li, K.-m. Chang, Y. Yuan, A. Hauptmann, Massive open online proctor: Protecting the credibility of moocs certificates, in: Proceedings of the 18th ACM conference on computer supported cooperative work & social computing, 2015, pp. 1129-1137.

[18] L. S. Nguyen, D. Frauendorfer, M. S. Mast, D. Gatica-Perez, Hire me: Computational inference of hirability in employment interviews based on nonverbal behavior, IEEE Transactions on Multimedia 16 (4) (2014) 1018-1031.

[19] A. K. Pandey, S. Kumar, B. Rajendran, B. Bindhumadhava, E-parakh: Unsupervised online examination system, in: 2020 IEEE REGION 10 CONFERENCE (TENCON), IEEE, 2020, pp. 667-671.

[20] E. Winarno, W. Hadikurniawati, I. H. Al Amin, M. Sukur, Anti-cheating presence system based on 3wpca-dual vision face recognition, in: 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), IEEE, 2017, pp. 1-5.

[21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), Vol. 1, Ieee, 2005, pp. 886-893.

[22] D. E. King, Dlib-ml: A machine learning toolkit, The Journal of Machine Learning Research 10 (2009) 1755-1758.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, C. L. Zitnick, Microsoft coco: Common objects in context, in: European conference on computer vision, Springer, 2014, pp. 740-755.

[24] Y. Li, H. Huang, Q. Xie, L. Yao, Q. Chen, Research on a surface defect detection algorithm based on mobilenet-ssd, Applied Sciences 8 (9) (2018) 1678.

[25] M. Danelljan, G. Hager, F. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: British Machine Vision Conference, Nottingham, September 1-5, 2014, Bmva Press, 2014.

[26] X. Liu, N. Krahnstoever, T. Yu, P. Tu, What are customers looking at?, in: 2007 IEEE Conference on Advanced Video and Signal Based Surveillance, IEEE, 2007, pp. 405-410.

[27] M. J. Reale, S. Canavan, L. Yin, K. Hu, T. Hung, A multi-gesture interaction system using a 3-d iris disk model for gaze estimation and an active appearance model for 3-d hand pointing, IEEE Transactions on Multimedia 13 (3) (2011) 474-486.