Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-20099-w

Auto-proctoring using computer vision in MOOCs system

Tuan Linh Dang[1] · Nguyen Minh Nhat Hoang[1] · The Vu Nguyen[1] · Hoang Vu Nguyen[1] · Quang Minh Dang[1] · Quang Hai Tran[1] · Huy Hoang Pham[1]

Received: 18 June 2023 / Revised: 30 July 2024 / Accepted: 7 August 2024

Abstract
The COVID-19 outbreak has caused a significant shift towards virtual education, where Massive Open Online Courses (MOOCs), such as EdX and Coursera, have become prevalent distance learning mediums. Online exams are also gaining popularity, but they pose a risk of cheating without proper supervision. Online proctoring can significantly improve the quality of education, and with the addition of extended modules on MOOCs, the incorporation of artificial intelligence in the proctoring process has become more accessible. Despite the advancements in machine learning-based cheating detection in third-party proctoring tools, there is still a need for optimization and adaptability of such systems for massive simultaneous user requirements of MOOCs. Therefore, we have developed an examination monitoring system based on advanced artificial intelligence technology. This system is highly scalable and can be easily integrated with our existing MOOCs platform, daotao.ai. Experimental results demonstrated that our proposed system achieved a 95.66% accuracy rate in detecting cheating behaviors, processed video inputs with an average response time of 0.517 seconds, and successfully handled concurrent user demands, thereby validating its effectiveness and reliability for large-scale online examination monitoring.

Keywords MOOCs · Online proctoring · AI proctoring · Computer vision

1 Introduction

During the COVID pandemic, educational activities have strongly shifted to online learning due to long quarantine periods. Massive Open Online Courses (MOOCs) offer a wide range of online learning functions, courses, and class management tools, and are optimized for massive learner access at the same time.

MOOCs are a type of online education that allows people to participate in online courses in large numbers. MOOCs are free online courses that give people from around the world a chance to learn. MOOCs usually include online lectures, exercises, and assessments, designed to ensure that learners can properly self-study. This is often done using online tools such as

Nguyen Minh Nhat Hoang, The Vu Nguyen, Hoang Vu Nguyen, Quang Minh Dang, Quang Hai Tran, Huy Hoang Pham These authors are contributed equally to this work

Extended author information available on the last page of the article videos, lectures, quizzes, and online discussions. The name MOOCs is derived from the characteristic features of this platform, which include:

- **Massive**: MOOCs accommodate hundreds to thousands of learners simultaneously, fostering a diverse learning community.
- **Open**:Accessible globally to anyone with an internet connection, MOOCs facilitate the sharing of knowledge and ideas worldwide.
- **Online**:Available anytime and anywhere via internet-connected devices, MOOCs offer convenience and cost savings for learners.
- **Course**: MOOCs provide structured learning experiences with lectures, exercises, and assessments, aimed at skill and knowledge development.

MOOCs system archives all the features mentioned above by using high-performance computing platforms over the Internet. There are several MOOC-based systems implemented worldwide, with notable examples such as Coursera developed by Stanford University, and edX developed by two leading universities in the world, Harvard University and MIT. These platforms offer online courses in many fields, like science, technology, economics, business, medicine, arts, and social science. They attract millions of students from different countries [1].

Along with online learning, online exams are also growing in popularity, and arguing their benefits over traditional methods. On the other hand, without an appropriate examination supervision method, online exams do come with a high potential risk of cheating. Furthermore, online proctoring during examinations had an enormous impact on students' behaviors, thus enhancing education quality. The possibility of integrating different modules into the MOOCs has enhanced the feasibility of implementing online proctoring using artificial intelligence (AI). Despite the potential benefits, AI modules are still not being fully utilized in online proctoring research. Even though using machine learning to recognize students' faces and developing third-party apps for online classes are crucial breakthroughs [2, 3], the current cheating detection through machine learning is inadequate, which is a fundamental necessity for proctoring systems. Additionally, these methods are not adapted to MOOCs and instead operate independently as third-party extensions. Atoum et al. [2] have proposed a cheat detection system that is adaptive with MOOCs. This approach succeeded in detecting fraud online, however, there are several limits to this system. Firstly, Using an SVM classifier that relies on audiovisual features from students' data can make it harder to discern evidence and details of fraud. Secondly, To collect audio and visual data from students, the system needs them to wear three devices: two cameras and one microphone. This can be uncomfortable for students during exams and also makes it expensive to provide the necessary equipment for large classes. A more streamlined approach to exam monitoring is being developed through ProctorU [4], which serves as a third-party application and reduces the need for multiple devices. However, this convenience does come with potential cybersecurity and privacy risks that must be taken into consideration. In addition, the installation of third-party applications by the student may cause a variety of issues such as incompatibility, privacy, and memory and storage problems due to their coexistence with the learning platform. These problems can be solved by developing a system that is adaptive or embedded into the learning platform to simplify the proctoring procedure.

With the recent blooming of Computer Vision achievements [5] and the rapid development of online learning platforms [6], online proctoring can be leveraged in various ways. Online learning management has opened up various methods of distance education, and research has shown that the most significant outcome of this trend is the skyrocketing popularity of MOOC platforms like Coursera, which are recognized by a countless number of universities and institutions. In Vietnam, MOOCs platform system named daotao.ai is being developed and operated by Edtech Centre, under the School of Information and Communications Technology of Hanoi University of Science and Technology, based on the open-source edX code. Up to now, the daotao.ai MOOCs platform has been successfully deployed for professional training institutions as well as community training support projects. As of March 2023, daotao.ai has over 20000 users with a total of 150 courses with many features designed to adapt the Vietnam context. MediaPipe, a framework for deep learning models, has introduced a lightweight face detection module that runs real-time inference on the CPU, making it highly efficient for use in the field of Computer Vision. Thanks to the advancements in hardware development, modern laptops are now capable of supporting a vast array of lightweight CPU models. Additionally, the evolution of webcams has resulted in sharper image quality and faster FPS rates, significantly enhancing the accuracy of inference. The utilization of GPU [7] has expanded the possibilities for proctoring, enabling the implementation of facial recognition to monitor attendance, object detection to identify unauthorized items, and gaze estimation. Undeniably, deep learning models have become more accessible in the field of online education, with increased precision and capability while being easier to implement.

This paper proposes an online proctoring structure that is adapted directly to MOOCs and performs automated, continuous supervising during an online exam. In supervision tasks, this system is capable of automatic attendance checks via face recognition, phone detection, and gaze estimation. By integrating directly into MOOCs as part of the exam, our proposed system simplifies the procedure and hardware requirements of online proctoring. Three key components make up the system: an AI server designed to handle multimedia requests, an examinee client integrated into MOOCs that records the examinee's video, and an examiner client responsible for monitoring the entire supervision process.

In summary, our main contribution is described below:

- We proposed an online proctoring structure to perform AI supervision that is adaptive to online learning platforms like MOOCs.
- We applied the proposed system to an actual MOOCs platform, daotao.ai, and evaluated its performance under actual conditions.

In the next section, we will review the related works to the proposed system, including other online exam supervision systems, AI techniques, and deployment services. The third section introduces the proposed system in detail for each component. The fourth section presents our evaluation environment and results, and the fifth section is the conclusion taken from the experimental result, outlining future improvement possibilities.

2 Related work

2.1 Proctoring systems

In recent years, with the reasons mentioned above, the importance of an exam system with monitoring support has gradually increased.

Many companies have developed exam support systems for commercial purposes and require payment to use these systems. Some typical systems include: SafeExamBrowser, ProctorU, and ProctorTracking.

SafeExamBrowser [8]: It is a computer software that limits access to only allowed websites and applications during the exam, preventing cheating and accessing unauthorized sources. It is often used by educational institutions for online exams. However, this system does not support video transmission to allow supervisors to monitor students during exams.

Mettl [9]: It is online testing platform that provides features such as customization, remote monitoring, real-time analysis, and AI-based monitoring through video, gaze analysis, and phone detection. This is a paid software and requires a registered account with a fee to use. However, the current system is not efficient for massive online proctoring since it is slow during live-streaming in exams and can only display one student at a time, which limits its ability to monitor multiple students simultaneously.

ProctorTracking [10]: It is an online monitoring system that applies AI to track students, ensuring that students are present in the video during the exam. The platform also allows students to share their screen to prevent cheating on the computer. However, using the system can be cumbersome and difficult to adjust to fit open-source learning management systems.

ProctorU [4]: It is an online testing platform that allows remote proctoring. The system is also supported by AI to assist proctors in monitoring with the feature of candidate face recognition. The system requires payment to use, and it is very difficult to customize to personal preferences. Also, the process is so complex and annoying, and it may not be able to handle many users.

Although many researchers have suggested methods to enhance exam monitoring, these approaches are still in the experimental phase and have not been implemented in practical settings yet.

Numerous techniques have been suggested to enhance the integrity of online exams, including both indirect monitoring systems and assistance for proctors, as well as AI-driven analysis systems. Wahid et al. [11] provided a website application and secure network design that can prevent cheating. Li et al. [12] also applied the MOOCs framework to implement both automatic and collaborative methods to detect cheating behavior in online tests, consisting of three components: the Automatic Cheating Detector (ACD), the Peer Cheating Detector (PCD), and the Final Review Committee (FRC). This is a semi-automated system using webcams and other sensors for monitoring. In addition, there exist alternative automated techniques for detecting test takers using several approaches [13]. These approaches involve the use of biometric data, such as facial recognition, voice recognition, and hand gestures, to identify test takers. Ganidisastra and Bandung [14] introduced a deep-learning system for facial recognition. In addition, there are many other studies to detect various cheating behaviors. Atoum et al. [2] applied AI and ML to integrate various cheating detection methods into the MOOC system such as user verification, text detection, voice detection, active window detection, gaze estimation, and phone detection. Although deep learning has been utilized by [15] to detect faces, blinking, and objects, the approach still falls short regarding speed and accuracy when dealing with multiple users. Despite the mentioned research, no widely utilized system has been developed and put into practice yet.

In brief, contemporary online proctoring systems provide a diverse collection of functionalities aimed at ensuring the integrity of examinations. Despite their varied capabilities, these systems exhibit several critical limitations that impede their overall effectiveness and integration with educational technologies, including:

- **Third-Party Application Status:** They operate as third-party applications that do not natively support integration with MOOCs. This lack of seamless integration complicates deployment and usage within the MOOC framework, leading to inefficiencies and potential compatibility and privacy issues.
- **Lack of Comprehensive and Automated Solutions:** Existing systems often fail to provide a holistic and automated approach to student proctoring. The absence of such

comprehensive solutions necessitates substantial manual oversight and intervention, which is labor-intensive and susceptible to human error.

- **Requirement for Additional Hardware:** Some proctoring systems necessitate the installation of supplementary hardware devices. This requirement imposes significant logistical challenges and inconveniences for both educational institutions and students, potentially acting as a barrier to widespread adoption.
- **Deficiencies in Accuracy and Real-Time Performance:** The use of naive algorithmic approaches in many AI-driven proctoring systems results in suboptimal accuracy and real-time performance. These deficiencies can compromise the efficacy of the proctoring process, thereby undermining the integrity and fairness of the examinations being monitored.

The diverse yet incomplete nature of these solutions underscores the pressing need for a unified and integrated proctoring system. Such a system should be capable of addressing the aforementioned limitations by ensuring native MOOC integration, offering comprehensive and automated proctoring solutions, eliminating the need for additional hardware, and leveraging advanced AI algorithms to enhance accuracy and real-time performance.

2.2 AI techniques in online proctoring

2.2.1 Face detection

In a face recognition system, face detection is typically the first step in the process. After detecting the faces in an image or video, the system becomes capable of extracting the facial features and comparing them with the database of recognized faces to identify or verify them.

The accuracy of face detection has significantly increased thanks to the advanced deep-learning models that are being used nowadays. Numerous cutting-edge deep neural networks have been purposely crafted and fine-tuned to discern exceedingly accurate human faces, while simultaneously ensuring the right balance between accuracy and speed. For example, MTCNN [16] was strong in terms of processing speed, while YOLOv5-face [17] was more versatile with backbones that focused on either speed or accuracy but lacked balance between these two.

RetinaFace [18] is a state-of-the-art face detection algorithm developed by researchers at the Chinese University of Hong Kong in 2019. It builds upon the RetinaNet framework and is a powerful single-stage face detector that executes pixel-wise face localization on various scales of faces based on combining both extra supervision and self-supervision in multitask learning. RetinaFace is able to improve the accuracy and efficiency of face detection in challenging scenarios. However, many parameters make the model run slower, which is not compatible with a real-time face recognition system. Such a problem can be overcome by using a lighter backbone, typicallyMobileNets.

MobileNets [19] is a class of lightweight models primarily intended for mobile and embedded vision tasks. These models use a technique called depth-wise separable convolution, which offers reduced computational requirements compared to regular convolutions, with only a minimal loss in accuracy. Depth-wise separable convolution involves two layers: depthwise and pointwise convolutions. Depthwise convolutions apply one filter to each input channel, while pointwise convolutions use a 1×1 convolution to merge the output of the depthwise layers linearly. This architecture has been instrumental in the success of MobileNets for efficient and effective computer vision applications.

2.2.2 Face alignment

After detecting the face in an image or video, the face recognition system identifies the human based on his or her face. However, in some situations, the result of the detection model may be inaccurate or difficult to verify. Therefore, it is necessary to use a face alignment module to deal with this problem.

Currently, there exist various techniques to address the challenge of face alignment which greatly enhance the efficiency of the recognition system. The first solution is to use the affine transformation [20]. This transformation is a popular 2D alignment method that can preserve points, straight lines, and planes. In addition to this, it can also change the rotation, the aspect ratio, and the localization of the shape. Besides that, some advanced deep-learning methods are applied to support the face alignment process, such as ATPN [21] or SLPT [22].

2.2.3 Facial feature extraction

Facial Feature Extraction is essential in an AI proctoring system, serve the purpose of extracting face images to vectors used in other use cases such as ID verification. Some neural networks could be used to classify the human face. However, those models need to be retrained if there are more people added to the system. Instead of attempting to classify a face that has been cropped from an image or video, utilizing matching techniques to locate the image of the most similar face in a database is a more efficient approach. Once a match is found, the human can be identified based on this image.

To implement matching solutions for face recognition, powerful feature extraction models such as ResNet, MobileNet, and EfficientNet are essential.

ResNet [23] is a well-known convolutional neural network that is designed to tackle the gradient vanishing problem when training a deep model. The main idea of this model is to use a residual block. This block contains a skip connection, which is directly connected to its output.

EfficientNet [24] is a famous lightweight architecture that Mingxing Tan and his partners introduced. In the paper on this model, the authors present the compound method, which scales the width, depth, and resolution of the neural network efficiently.

2.2.4 Feature matching

K-nearest neighbor (KNN) is a lazy-supervised learning algorithm that does not need to be trained and only calculates in the inference phase. In AI proctoring, Feature Matching usually used in mapping a face feature of a student to another to determine whether they are of the same person. This model predicts new data based on the labeled data by finding the k-nearest neighbor of that data. The Euclidean equation measures the distance between the data to predict and the data in the database.

FAISS (Facebook AI Similarity Search) is an open-source library developed by Facebook AI Research that specializes in efficient similarity search and clustering of high-dimensional vectors. FAISS is particularly suited for tasks such as searching through large databases of feature vectors. This method is able to handle large datasets with many dimensions and can scale accordingly. It is ideal for managing very large datasets that traditional methods cannot handle. Additionally, it provides a collection of algorithms and data structures to perform similarity searches on the CPU and GPU efficiently. Due to its capabilities, FAISS is commonly utilized in tasks involving the retrieval of images or text.

2.3 Backend service for proctoring system

Almost every functional software system would require at least one or even a group of components that acts as a database system. The application needs a database to keep track of its actions and customers and to meet their needs. Without it, the application cannot record user behavior or usage information.

For recording and exporting attendance reports of users' classes, a simple but sufficient backend system was deployed with the following main components:

MySQL Database is a popular relational database management system (RDBMS). RDBMS has pros and cons. Compared to other NoSQL databases, it is less flexible and needs better schema-designing processes. However, these limitations also have benefits. RDBMS is highly preferred over other types of databases because of fewer unexpected behaviors and uniform data formats. This, in turn, provides reliable performance for the users and results in better data management.

Spring Boot Application acts as a middleman between the clients and the database. Spring Boot itself is an open-source framework written in Java. Its flexible architecture enables people to spin up a microservice in a short time.

TensorRT is a high-performance deep learning inference optimizer and runtime library developed by NVIDIA. It provides tools and APIs to accelerate the development of neural networks on NVIDIA GPUs, including those used in devices and data centers. TensorRT archives this acceleration by performing several optimization techniques, such as layer fusion, precision calibration, kernel auto-tuning, and dynamic tensor memory management. These optimizations result in lower latency and higher throughput for inference applications. TensorRT supports various network architectures, including CNN, RNN, LSTM, and Transformer.

3 Proposed system

3.1 General

Our proposed solution is a monitoring system for online exams which can automatically authenticate the examinee's identity and accurately detect their facial orientation. Additionally, it has the ability to detect if the examinee is attempting to use a phone during the exam. It is built adaptively with the MOOCs platform, with student clients operated based on EdX's xblock and able to integrate into MOOCs' examining section. The proposed structure aims to improve the field of online proctoring by utilizing AI applications on MOOCs platforms. The goal is to maintain high speed and openness requirements while taking advantage of AI technology. To accomplish the preceding features, the system involves four main modules: the examinee client, the examiner client integrated with the MOOCs server, the AI server, and the backend services as can be seen in Fig. 1.

The AI Server stands at the core of this system, entrusted with the crucial task of accepting multimedia streams from the student client, analyzing them to generate proctoring outcomes, interacting with the student client, and storing the results in the database. This module also handles client queries regarding proctoring results and provides them with timely responses. To perform such features, AI Server includes lightweight deep learning models capable of real-time processing.

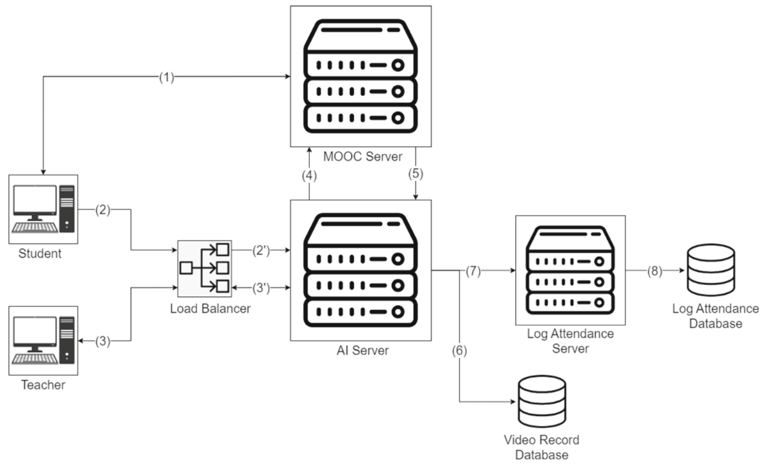


Fig. 1 System architecture

MOOCs Server is the big platform that hosts various services for online learning and course management. Our system does not include an MOOCs server component, but it is used as the foundation for integrating the examinee and examiner clients. The MOOCs server is a platform that offers valuable data about students and courses to both clients and the AI server for administrative purposes.

The Examinee client plays the role of student interface. This module is embedded as a unit of a MOOCs' exam via xblock and will display a front camera view of a student, showing him or her being proctored and his or her proctoring results. The Examinee module receives inputs like permission for multimedia access, camera media streams from student laptops, and students' public information such as ID numbers. The output of this module is the multimedia stream sent to the AI server for processing. The client will connect the proctoring system and the MOOCs platform. This client is significant because it is the common part between the two systems. This facilitates the exchange of information and interaction between both systems.

The Examiner client is the teacher interface for monitoring the proctoring system. With this module, teachers can decide when to start or stop the attendance check and query for results immediately. This module takes input from an exam's information about the students' list and ID, course ID, and timestamp when the exam happened. When the teacher executes an action, this client sends out a request, delivering the necessary information to the AI server to monitor the exam.

The Backend services offers a range of deployment features aimed at efficiently managing data flow from clients to the AI server, enhancing server concurrency, and storing AI server outcomes to a database. This module also configures the connection and API communication among student clients, teacher clients, and the AI server. These services may leverage the server's performance in terms of response time and concurrent processing ability.

3.2 AI server

The AI server delivers an array of AI services that are designed to curb cheating during exams. The services include attendance verification, checking for head pose and phone usage, and detecting drowsiness. This keeps students focused and minimizes cheating.

3.2.1 Attendance checking service

Our proposed attendance checking system was designed to be an end-to-end face recognition system. The whole process pipeline can be seen Fig. 2.

The system consisted of four modules: (i) Module **(A)** was responsible for detecting faces and five facial landmarks from raw video frames; (ii) Module **(B)** performed the face alignment process; (iii) Module **(C)** then extracted the feature embedding from aligned faces; (iv) Finally, module **(D)** was implemented with a feature matching strategy to produce the final Face IDs as the output.

The proposed system utilized RetinaFace [18] with MobileNet backbone in the face detection module (A) to achieve improved accuracy for both face detection and landmark localization without compromising the speed. In the face alignment module (B), the five landmarks detected in the previous stage were used to align faces using affine transformation. In the facial feature extraction module (C), the model provided by face.evoLVe [25] became a suitable choice since it was fine-tuned on an Asian face data set. More specifically, this model had its backbone IR-50 trained on the Ms-Celeb-1M dataset [26] using focal loss and then fine-tuned on face.evoLVe private Asian face dataset using ArcFace [27] loss. In the proposed system, the model was converted to TensorRT for faster inference speed. Lastly, in the feature matching module (D), FAISS-adapted k-NN was used for optimizing the speed when matching on a large-scale feature embedding database.

In the upcoming sections of this chapter, a detailed description of each module will be provided. The process of constructing the dataset will be outlined in Section 4.2, Datasets.

(A) Face Detection & Facial Landmark Localisation:

Out of all the face detection and facial landmark location models available, the RetinaFace model stood out as the best choice for our proposed system, thanks to its implementation of a MobileNet 0.25 backbone in deep learning. This combination provides decent accuracy with a much faster processing speed in comparison with the ResNet backbone.

(B) Face Alignment:

The proposed system uses an affine transformation to align face images. Affine transformation is a 2D alignment method that preserves points, straight lines, and planes. Examples

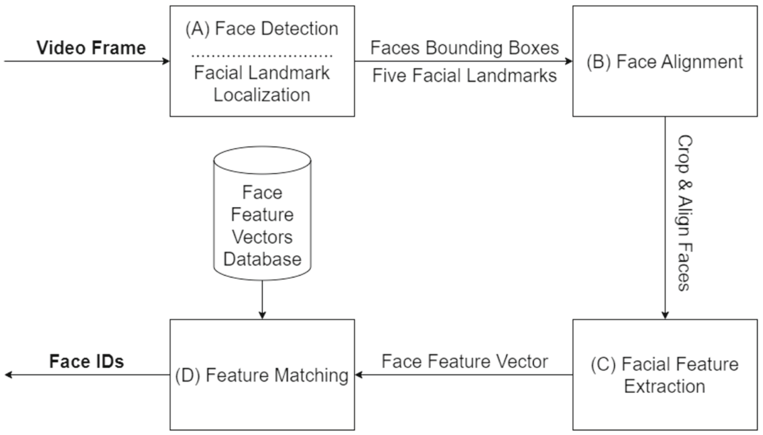


Fig. 2 Attendance checking pipeline

of affine transformations include translation, rotation, scaling, changing the aspect ratio, and shear mapping. An example of affine transformation can be seen in Fig. 3.

(C) Facial Feature Extraction:

To correctly identify Vietnamese faces, a strong deep-face feature model trained with Asian face images was crucial. As mentioned in the earlier section, the proposed system was developed based on face.evoLVe [25]. The main reason was because of the face.evoLVe provided a deep model that matches the criteria. The model used in this feature extraction module has an IR50 backbone that was pre-trained with Focal Loss on the Ms-Celeb-1M [26] dataset. The key factor is that this model was then fine-tuned on an Asian data set with ArcFace loss. However, face.evoLVe only provided the original model without any attempts to speed up the inference time. To reduce the required time for inference, the model has been converted into a TensorRT version. With the TensorRT version of the IR50 Arcface model, the feature extraction module in the proposed system had been able to process with less time required and a negligible accuracy drop.

(D) Feature Matching:

The feature-matching module plays an important role as the last module in the whole system. This module directly affects the results of the proposed system. The system needed to work on a large database, so the methods used in the feature-matching module had to be quick, precise, and effective. k-NN can guarantee accuracy and stability, while the FAISS framework can help k-NN search much faster. Therefore, k-NN implemented with FAISS became the solution in the feature matching module.

The proposed system utilized k-NN FAISS, which provided a list of the k nearest neighbors, or the k feature vectors with the closest distance to the query feature vector. We proposed **Two-stage Threshold Algorithm**-a simple voting mechanism used to decide which class or which identity the query face belongs to. The class ID that has the largest number of vectors close to the query feature vector was selected as the output of the voting step. The algorithm prevents the feature matching stage from identifying a face image with a non-existent ID in the database. Overcoming this situation is easily achievable by implementing a distance threshold, which is a simple and effective solution. Once the ID has been generated in the previous step, it is possible to calculate the distance between the query feature vector and the closest vector associated with that specific ID. Let's denote that distance with the symbol d. Afterward, the value of d is compared to a pre-defined distance threshold, D. If d is smaller or equal to D, then the output ID is the actual ID that the query face belongs to. On the other hand, if d is greater than D, the output will be "Unknown", which means the query face does not belong to any IDs in the database. Figure 4 illustrates this process flow.

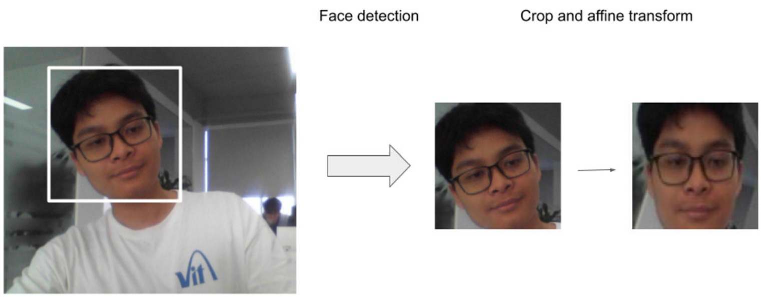


Fig. 3 Crop face and affine transform

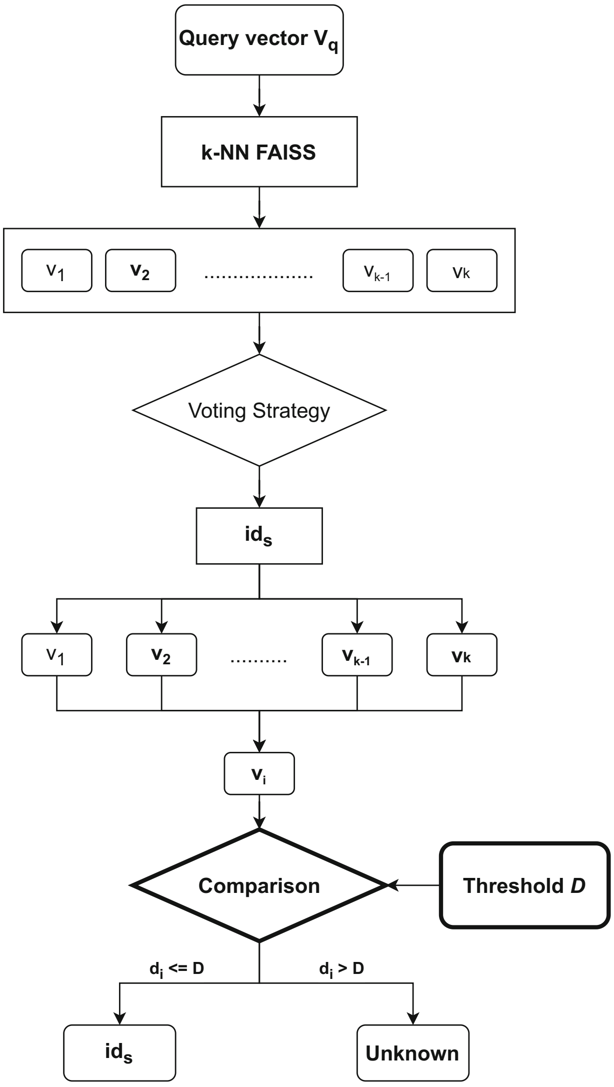


Fig. 4 Normal distance thresholding

The system might not be entirely accurate when working with video streaming as an input. The reason for this is that fast-moving objects or unfavorable lighting conditions at specific angles might cause blur, leading to a decrease in precision. A possible scenario is when a facial recognition system keeps oscillating between correctly identifying a face and mislabeling it as “unknown”. A blurred face makes its information less clear, which makes it harder to match with clear faces in a database. Two problems are causing the issue. Firstly, the faces in the video are not pre-processed before being fed into the feature extraction model. Secondly, the normal method of setting a distance threshold isn’t very effective when dealing with video streaming. Several pre-processing methods, such as Richardson-Lucy deconvolution or Point Spread Function (PSF) were examined and implemented. Facial recognition technology did not improve much because it is hard to see important facial features such as eyes, nose, and lips in small videos. Classic image sharpening methods are not effective for accurately recovering these features. Previous studies have proposed several methods for enhancing blurry facial images. However, the majority of these approaches relied on the use of deep learning models as the solution. Hence, these methods required more time to process face images. The system would also need stronger memory and graphics to operate with its two deep-learning models for detection and recognition.

To minimize the negative effects of motion blur, the proposed system was implemented with a new distance thresholding procedure called a *two-stage thresholding algorithm* that made use of previously recognized faces in a video. Figure 5 depicts the flowchart of this algorithm.

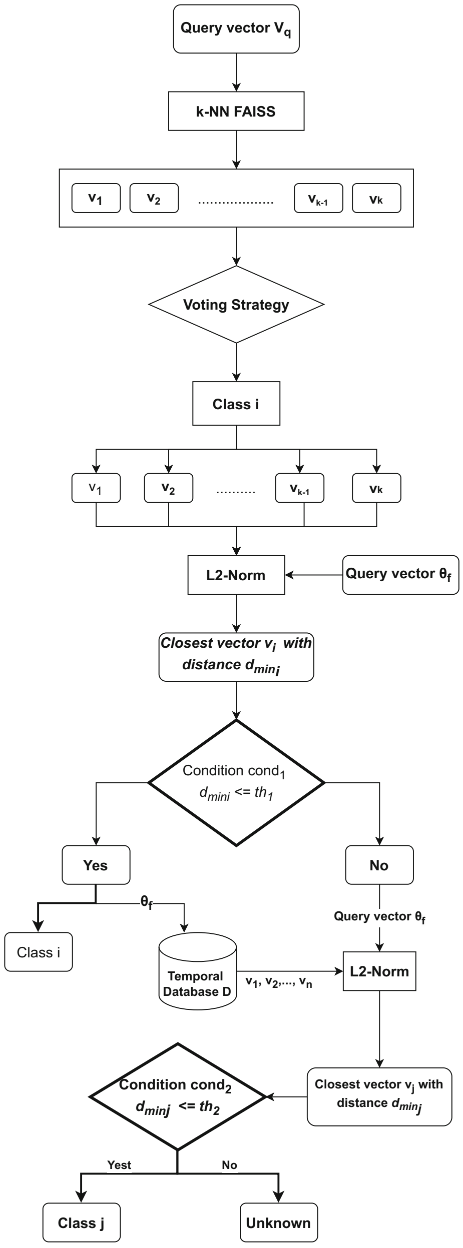
The intuition behind the new threshold algorithm can be described as follows. During the processing time, for each θ recognized at a moment m, if the distance d_{min_i} between feature θ_f of face θ and the closest feature vector v_i of class i (class i is the result after k-NN FAISS and voting procedure) is smaller than threshold th_1 (condition $cond_1$), the class i will be decided as the output of the whole system and the face feature θ_f will be stored in a temporal data structure $\mathbb{D}$. On the other hand, if d_{min_i} is larger than th_1, the feature vector θ_f will be compared with every feature vector stored in $\mathbb{D}$, and if the closest distance d_{min_j} than a threshold th_2 (condition $cond_2$), the corresponding i stored alongside with the vector in $\mathbb{D}$ will be returned, or else label “Unknown” will be returned. For the following frames, whenever the first condition $cond_1$ is satisfied, $\mathbb{D}$ will be updated by taking the average of all previous vectors with the new one θ_f. In other words, $\mathbb{D}$ stores different IDs with their corresponding average feature vectors that are computed from all quality feature vectors that appeared during the time. $\mathbb{D}$ can be considered a temporal backup database to do feature matching whenever the original database fails to do so. With the appearance of a temporal backup database $\mathbb{D}$, the proposed system suffers less from the blur phenomenon. The reason for this is that $\mathbb{D}$ can retain the high-quality feature embedding of the face before it starts moving quickly, thereby preserving specific facial features before they start to fade. Despite the movement and blurriness of the face, there may still be unique characteristics that can be utilized for identification purposes by extracting feature vectors from previous frames.

3.2.2 Phone detection service

The Phone Detection service was made to prevent cheating during exams and detect phone use, in addition to the Attendance Checking service. This module will be running until the end of the contest. The module is designed to prevent cheating during tests by using an AI model to identify phone usage by students. If the phone is detected, it alerts the teacher with a cheating notification and saves the illegal video to a database for future reference. YOLOv10 [28] which is an end-to-end object detection model was adopted as a phone detection model. This model was chosen because it is achievingstate-of-the-art results in the object detection problem in both accuracy and processing speed. The operation of this service is presented in Fig. 6.

The phone detection module was built based on YOLOv10 which is the state-of-the-art model in object detection. YOLOv10 stands out for its ability of precisely detecting small objects as well as real-time processing at high speeds and effectively high-resolution images processing.

Fig. 5 Two-stage distance thresholding



We use the YOLOv10s version to optimize program runtime. Through experimental comparison between 6 versions of YOLOv10, we found that the YOLOv10s version has approximately the same accuracy as the 3 higher versions YOLOv10b, YOLOv10l, YOLOv10x. However, YOLOv10s has lower processing time so it will be suitable in our problem. The comparison results are shown in section 4.3.1

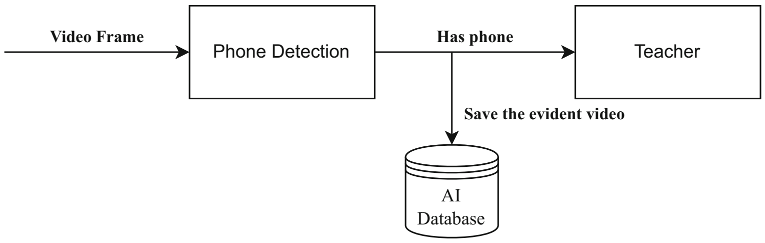


Fig. 6 Phone detection system

3.2.3 Head pose checking service

In computer vision, the pose of an object refers to its relative orientation and position with respect to a camera. The service aims to make sure the student's face is facing the screen during an exam. To do this, we built a head image processing pipeline to determine the direction of head rotation and classify whether that rotation represents an act of exam cheating or not. The pipeline is shown in the Fig. 7. The gaze recognition model will be processed sequentially through three modules: (A) Broadcast face display, (B) Determine three head rotation angles, (C) Classify gaze direction.

In part (A), the RetinaFace model is adopted as a face recognition model. In part (B), the HopeNet model [29] is used to determine the three angles head rotation (Yaw, Pitch and Roll). Finally in part (C), three head rotation values(Yaw, Pitch and Roll) will be obtained passed through an SVM classifier to classify whether that gaze direction is normal or not suspect. The Support Vector Machine (SVM) algorithm was chosen due to its suitability for classification tasks. SVM's tolerance for noise and errors in data makes it well-suited for classifying head pose based on head angles. Additionally, SVM models are known for their efficient processing speed so we can optimize our system performance.

3.3 Backend service

This section is about non-AI services, including Load Balancing and Attendance Logging services.

3.3.1 Load balancing service

Since Python only works in single-threaded mode [30], a solution to handle multiple concurrent users' requests needs to be implemented. The service has two important functions: acting as a reverse proxy and distributing workloads between AI server instances.

Reverse Proxy [31] helps hide the addresses of actual AI servers, mitigating the damages of possible attacks on the system. It also provides a unified access endpoint for every worker's servers, which simplifies the integration process for other developers.

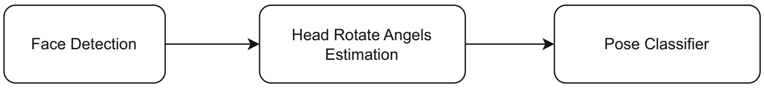


Fig. 7 Head pose classifier pipeline

The service also acts as a workload distributor. The Round Robin algorithm distributes user requests across many servers instead of overloading one server. This approach prevents an endless queue of requests, which can cause service disruptions. The illustration of the service can be seen in Fig. 8.

3.3.2 Attendance logging service

The AI server requires a storage location for the results obtained after receiving and processing user videos, which can be used to extract attendance check reports, and can also help in enhancing the model in the future by studying the results obtained from users. The system consists of two smaller modules at can be observed in Fig. 9 and as follows.

- A RESTful [32] web service using the Spring Boot Framework that provides accessible APIs
- A MySQL database for storing data from the RESTful web service

3.4 Student monitor module

The student monitoring module is designed to capture real-time images of students during exams. This is to ensure academic integrity by detecting any attempts at cheating or unauthorized behavior. The module is organized as a unit of MOOCs' exams and is constructed from a front-end interface and a back-end monitor.

The module's interface has a video window that shows live footage from the camera and a panel that displays the student's name, ID, and the monitoring results during the exam. The live video window is essential to our monitoring system because it helps students adapt their seating positions to match the camera angle. This, in turn, makes it easier for the system to recognize each student accurately. Additionally, the real-time monitoring results displayed on the panel create a sense of accountability for students, which can reduce the likelihood of cheating during exams.

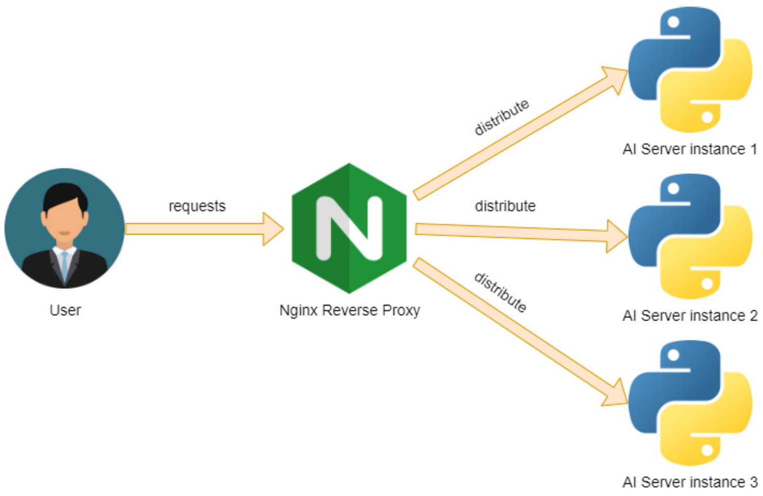


Fig. 8 Load balancing service diagram

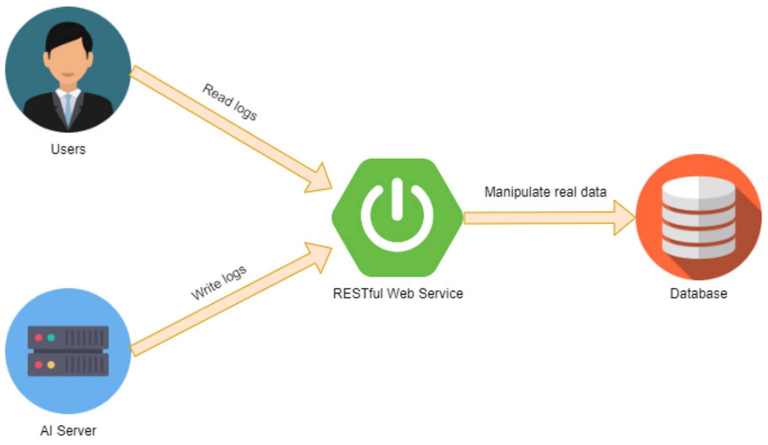


Fig. 9 Attendance logging service diagram

The back-end component monitors the mechanism of sending multimedia data from the webcam to the AI server and receiving proctoring results. The students' videos are recorded and sent to the AI server while also being streamed on their exam windows. To fulfill the objective of massive concurrent users, the workload on the internet must be handled carefully. The module records short video segments, splits them into smaller parts, and sends only the first segment of every period to the AI server. This is to reduce the impact on network bandwidth and storage capacity. The video is transformed into a concise string format and compressed to avoid overloading the AI server and ensure smooth transmission. The server will then process them using artificially intelligent algorithms to identify any irregularities or suspicious activity.

It is difficult to identify students who are sending network packages for one exam apart from those who are sending packages for another exam at the same time, when there are multiple users. This problem happened due to the registration of students being done before the exam on MOOCs' courses. To manage this situation, the back-end component is developed on the basic unit Xblock of the edX platform, which is also the basic unit for developing MOOCs. This particular element is linked to the server of MOOCs, establishing a direct connection to gather significant data about students, courses, and exams. It is then transmitted alongside the video content packages. The AI server uses this information to identify the student in the incoming video. After processing the video, the AI server will send the results back to the student monitoring module, which then displays the information to the test-taker. The results include student recognition, drowsiness detection, phone usage detection, and sitting posture recognition. Moreover, the system keeps track of all the detected irregularities in a database for future review and investigation, if necessary.

3.5 Teacher module

We have also introduced an interface for supervisors to monitor students' exams. Supervisors can use the server to execute built-in AI functions like face recognition, phone detection, drowsy detection, and face pose detection on specific students.

Initially, the server will not check the attendance of students until receiving a request from the proctor. The teacher will interact with the server via an API. When a request is received, the server will use the AI face recognition module to mark the attendance of specific students. Additionally, the supervisor can manually double-check all students' attendance during the exam to prevent any unauthorized substitution of exam takers. The semi-automatic interface design is highly effective in stopping academic misconduct and lowering the server's computational workload.

Moreover, the interface empowers supervisors to examine the outcomes produced by each AI module, in conjunction with the relevant evidence that the system highlights as a likely breach of exam regulations.

4 Experiment

4.1 Environments

Hardware: Our system's hardware setup was specifically designed to accommodate the considerable computational power required by AI modules, which necessitated extensive parallel processing capabilities and the ability to efficiently handle concurrent requests from MOOCs clients. Hence, the AI server was equipped with powerful graphic cards and processors. The details of the hardware environment are reported in Table 1.

Software Dependencies: For the implementation, the AI modules required several important dependencies:

- PyTorch version 1.13 and OpenCV 4.6.0 for the basic implementation of the face recognition module and phone detection module.
- Tensorflow version 2.11 and TensorRT 8.6.0 for optimizing the inference models.
- CUDA 11.7 and CuDNN 8 for running with GPUs.

4.2 Datasets

It is necessary to have a dataset for face recognition using the attendance checking system. Our experiments utilized two datasets to evaluate the effectiveness of AI algorithms: one for face recognition in attendance tracking, and the other for detecting fraudulent actions through phone detection. The construction of the face recognition dataset can be observed as follows.

- **Collecting Strategy**: The main goal of this step is to build a dataset for class-required attendance checking. To collect facial images from many individuals with maximum efficiency, the approach taken for data collection was as follows: Each individual was seated in front of a laptop equipped with an high-definition webcam. Then he or she was asked to slowly rotate his or her head while the camera consecutively captured photos.

Table 1 Hardware environment of AI Server

	Hardware
Main processor	AMD Ryzen 3.8Ghz 24 cores
Total memory	64GB DDR4
GPUs	2x NVIDIA Quadro RTX A4000 16GB
Hard drive	1 TB

- **Photos Requirements**: For each subject, the collected images must meet the following requirements: The photos need to be high-resolution (at least 720 × 720), taken with good lighting, and show the face clearly. Five images should simulate different poses at angles including the direct frontal face, two sides of the face, and an up-down angle.
- **Images Processing**: Once we gathered all the original facial images, we applied three different modules to enhance them: face detection and identification of facial landmarks, face alignment, and quality assessment. The final result consisted of face pictures that were sized at 112 × 112 and perfectly aligned. The details of the process can be seen in Fig. 10.

To assess the AI modules and the overall system's efficiency, we gathered two distinct datasets that serve two primary purposes: the first is to recognize faces through attendance records, and the second is to detect fraudulent activity through phone detection. The dataset for phone detection was obtained similarly to the face recognition dataset. Both datasets were captured through the MOOCs exam module to ensure that the data used in real-world scenarios on the AI server is accurately represented. Details and statistics of two datasets are described in Table 2.

4.3 Results

Given that the proposed structure processes input from a short video to generate results for each model accordingly, we assessed the processing time of each model on **a per-frame and per-video** basis to evaluate performance. Faster processing times, while maintaining acceptable accuracy metrics, indicate superior system performance.

Additionally, we conducted **end-to-end performance** evaluations to ensure comprehensive assessment of the proposed system. This metric encompassed the entire processing workflow, including the time taken during the inference phases of all AI models and the time required for video and result transmission over the network. Specifically, we measured the duration from the initiation of video transmission to the server to the moment the result was received by the user's browser.

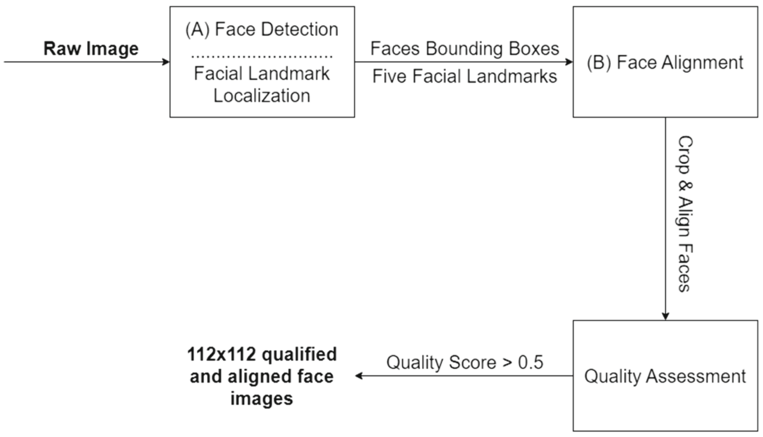


Fig. 10 Face image processing

Table 2 Face detection and phone detection datasets statistics

Dataset	No. videos	Avg. video duration (second)	No. classes	FPS	Format
Phone detection	2595	0.5	2	14	MP4
Face detection	4311	0.5	7	30	MP4

Finally, we assessed the **accuracy metrics** of the AI models to evaluate how well the proposed system maintains overall AI performance while optimizing processing efficiency. For this purpose, we calculated the accuracy metrics of each model during the aforementioned processing performance experiments.

We carried out the experiment in two distinct scenarios. Initially, we assessed the efficiency of the stand-alone AI components, followed by evaluating the efficiency of the system that incorporates MOOCs student clients. The purpose of this discrimination was to evaluate if the system was adaptive enough when applied to MOOCs.

4.3.1 AI modules evaluation

In this experiment, we did not need to find a method to collect images of students because we utilized pre-trained weights and existing datasets to train all the models. We evaluated the capability to detect faces, detect phones, and recognize faces. The weights for the face detection model were sourced from RetinaFace [18], and the weights for the face recognition model were derived from face.evoLVe [25], both trained on the MS-Celeb-1M dataset [26]. The weights for the phone detection model were obtained from the YOLOv10 model [28]. We assessed face detection using the VN Celebration dataset, which includes 23,105 images of 1,020 Vietnamese celebrities [33]. As this was just an initial experiment, we conducted tests internally within our research group; all the photos presented in this manuscript feature the study's authors.

Facial recognition and phone detection utilize distinct techniques influenced by their respective algorithms and the specific contexts of their applications. The face recognition process in videos was more complex. First, OpenCV read the video frames, and then features were extracted from each frame and compared to the face dataset. After that, the most frequently detected name was determined by taking a vote from the frames. The phone detection dataset included short videos captured in OpenCV. We evaluate by measuring whether each video contains a phone-detectable frame compared to a set of hand-labeled labels. To detect cheating on phones during exams, students' behaviors during the scenario were similar to actual cheating behaviors. We evaluated 6 versions of YOLOv10 (n,s,m,b,l,x)

Table 3 Phone detection YOLOv10 comparison result

Model	Accuracy	Average processing time (ms)	GPU utilized (MiB)
YOLOv10n	0.87	118.21	549
YOLOv10s	0.93	127.14	581
YOLOv10m	0.93	140.84	603
YOLOv10b	0.93	183.78	633
YOLOv10l	0.93	190.15	641
YOLOv10x	0.93	288.76	667

Table 4 Face detection and phone detection result

Module	Avg. video processing time	Avg. frame processing time	Accuracy
Phone detection	0.127	0.004	93.1%
Face detection	0.07	0.004	99.9%

which differ in the number of parameters with the datasets mentioned above with the criteria of accuracy, running time, and amount of GPU utilized to choose the suitable YOLOv10 version to deploy to production. The result is shown in the Tabel 3 We evaluate both the face recognition model and phone detection on GPU runtime. Accuracy and time process measurements in this circumstance can be observed in Tables 4 and 5. This comparison table shows that the YOLOv10n version has the same predictability as the heavier versions and has the fastest running speed. So we use this version of YOLO for phone detection model.

4.3.2 System evaluation

We conducted an experiment to apply AI models to MOOCs and measured their performance. We asked seven participants to take a mock exam on the MOOCs platform and recorded their network response time and recognition accuracy. This was how the system operates: customers created videos, encoded them, and then transferred them via the internet to the AI server. Then, they waited for the server to respond with the result. The AI server got videos from students, detected phones and faces, recognized faces, and sent the outcome back to the clients. The response time was calculated from the time the video was sent until the result was received. The results measured in the second situation are shown in Table 6.

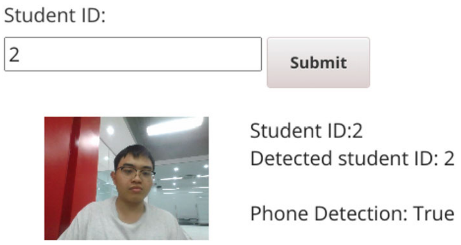
Additionally, this circumstance also tested the workings of the student client, load balancer solution, and head pose estimation warning. The AI server showed the exam supervision result on a screen to let students know they were being monitored during the test. The client example is shown in Fig. 11. The teacher client interface is shown in Fig. 12. The head pose estimation module was also involved in proctoring progress. The example of the head-pose estimation example can be seen in Figs. 13 and 14. In addition, the installation of the Nginx load balancer feature resulted in a remarkable enhancement in the overall system performance regarding the number of simultaneous users and response duration. With the integration of the Nginx service, the system could serve multiple concurrent users at the same time. In our evaluation circumstance, the system was tested on 10 concurrent accesses.

Table 5 Face recognition result

Class	Avg. video processing time	Avg. frame processing time	Accuracy
0	0.34	0.05	99.3%
1	0.6	0.03	88.7%
2	0.33	0.04	96.0%
3	0.96	0.03	85.7%
4	0.85	0.03	99.6%
5	0.25	0.05	87.6%
6	0.37	0.04	93.9%

Table 6 MOOCs server performance

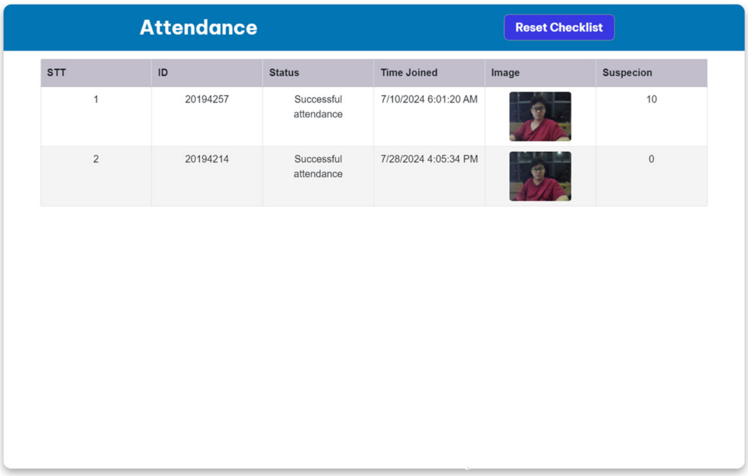
Total videos sent	Avg. response time (seconds)	Accuracy
4311	0.517	95.66%



Student ID:2
Detected student ID: 2

Phone Detection: True

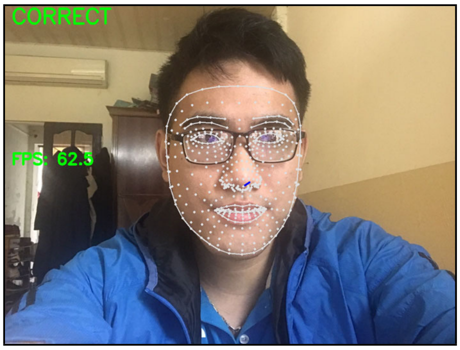
Fig. 11 Student client interface



STT	ID	Status	Time Joined	Image	Suspecion
1	20194257	Successful attendance	7/10/2024 6:01:20 AM		10
2	20194214	Successful attendance	7/28/2024 4:05:34 PM		0

Fig. 12 Teacher interface

Fig. 13 Correct face pose



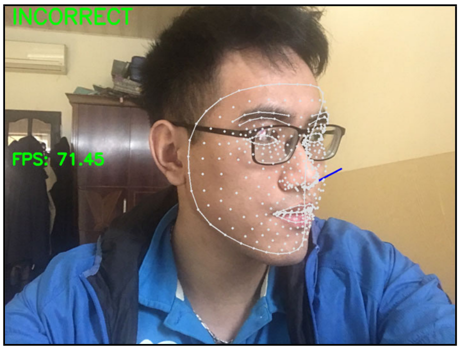


Fig. 14 Incorrect face pose

In Table 7, we measured the GPU memory usage of each model in the end-to-end circumstance. Overall, all three models for face recognition, face detection and phone detection summed up to 5.4 GB of GPU memory, and can be efficiently fit to one single GPU to serve separately.

Based on the assessed performance metrics, including both processing power and accuracy, our system has demonstrated its efficacy in integrating artificial intelligence models with the MOOCs system to effectively monitor online examinations. The proposed online proctoring system offers a comprehensive suite of features that distinguish it remarkably from other existing systems. Unlike its counterparts, which primarily focus on either:

- Basic functionalities such as browser lockdown [8], or
- Heavy dependence on supporting devices like cameras and voice recorders [12], or
- Separate development as third-party paid applications, or
- Naive machine learning/AI algorithms for detecting fraudulent behaviors

Our system has successfully addressed the downsides while enhancing proctoring quality by integrating advanced artificial intelligence algorithms to provide real-time behavioral analysis and anomaly detection with flexibility and natively support MOOCs platforms while preserving data security during examinations. In summary, the features our proposed system has thrived compared to others are described in Table 8. The comparing features are listed in rows, and the columns whether a proctoring system can provide the corresponding feature. The value of each cell is indicated as below:

- "Y" - if a system is capable of providing the row-corresponding feature or require the information to correctly function.
- "NP" (Not providing) - if a system is not capable of providing the row-corresponding feature
- "NR" (Not required) - if a system does not require the corresponding information

Table 7 System GPU resource usages

No.	Model	GPU memory usage
1	Face recognition	2 GB
2	Face detection	2.8 GB
3	Phone detection	0.6 GB
Total		5.4 GB

Table 8 Online proctoring systems features comparison

Feature	Proposed system	Safe exam browser [8]	Mettl [9]	ProtorU [4]	ProtorTrack [10]	AiAP [34]	Image-hasing-based [3]	Proctor SU [35]
Basic functionalities								
Internet required	Y	Y	Y	Y	Y	Y		Y
Taking user browser control	NR	Y	Y	Y	Y	Y	NR	Y
Fully Automated	Y	NP	NP (Semi automated)	NP (Semi automated)	Y	NP (Semi automated)		Y
Report generation	Y	NP	Y	Y	Y	NP	NP	Y
Anomaly behaviors loggedY for later consideration	NP	Y	Y		NP	NP		
MOOCs integration features								
Natively embedded in MOOCs	Y	NP	NP	NP	NP	NP	NP	NP
Is third-party application	NR	Y	Y	Y	Y	Y		Y
Require additional installation rather than MOOCs	NR	Y	Y	Y	Y	Y	Y	Y
Privacy-related features								
Take over user device control	NR	Y	Y	Y	Y	Y	NR	Y
Third-party information transmission	NR	Y	Y	Y	Y	Y	NR	Y

Table 8 continued

Feature	Proposed system	Safe exam browser [8]	Mettl [9]	ProtorU [4]	ProtorTrack [10]	AiAP [34]	Image-hasing-based [3]	Proctor SU [35]
Use third party API	NR	NR	NR	NR	NR	Y	NR	NR
External device requirements								
Webcam required	Y	NR	Y	Y	Y	Y	Y	Y
Voice recorder required	NR	NR	Y	NR	Y	Y	NR	Y
Other hardwares required	NR	NR	Y	Y	Y	NR	NR	Y
AI features								
Is AI-based application	Y	NP	Y	Y	Y	Y	Y	Y
Apply state-of-the-art AI algorithm	Y	NP				Y	NP	Y
ID verification	Y	NP	Y	Y	Y (Under upgraded)	Y	NP	Y
Phone detection	Y	NP		NP		Y	Y	Y
Anomaly behavior detection	Y	NP	Y	Y	Y (Under upgraded)	Y	Y	Y
Headpose estimation	Y	NP		NP		Y	NP	Y
Realtime analysis	Y	NP	Y (Under upgraded)	Y (Under upgraded)		NP	Y	Y

- “” (Blank) - if the necessary information is missing, insufficient to conclude or not mentioned.

5 Conclusion

In this paper, we have proposed an online proctoring system to perform examination supervision adaptively using a MOOCs platform. The system includes four main components: the AI server, student clients, teacher clients, and backend service. The system employs deep learning technologies to detect phones, recognize faces, and estimate head pose. An AI server analyzes the video feed from student clients to monitor attendance and behavior during exams. The teacher client is a necessary tool to manage exam classes effectively. By incorporating backend service, the system receives a boost in both performance and security. Experimental results showed that the proctoring system was built successfully. The performance was also assessed with datasets obtained from real participants. Furthermore, the system was integrated into a MOOCs platform, daotao.ai, to examine the interface and effectiveness of AI proctoring in actual exams. The results of our study on online proctoring revealed a highly encouraging design that can be effortlessly customized for different MOOC platforms. This design boasts quick response times and guarantees accurate deployment of AI modules.

As a prototype, we plan to conduct future research to test the functionality of our proposed systems with actual classes and students. In addition, models may have problems performing well in low-light conditions. Addressing this issue in future research is crucial, as the lighting conditions in examination rooms are only sometimes optimal. Phone detection can lead to misidentifying other objects that have a similar shape to phones, like calculators. This issue should be addressed in the future.

Acknowledgements This research is funded by Hanoi University of Science and Technology (HUST) under grant number T2022-PC-052. This research is also partially supported by NAVER Corporation within the framework of collaboration with the International Research Center for Artificial Intelligence (BKAI), School of Information and Communication Technology, HUST under project NAVER.2022.DA02.

Data Availability Accessing our dataset is available from the corresponding author upon reasonable request.

Declarations

Conflicts of interest The authors have no conflicts of interest to declare.

References

1. Vaidya S, Paranjape A (2014) Moocs–changing the way of education. In: 2014 IEEE International Conference on MOOC, Innovation and Technology in Education (MITE), pp. 362–365. https://doi.org/10.1109/MITE.2014.7020304
2. Atoum Y, Chen L, Liu AX, Hsu SDH, Liu X (2017) Automated online exam proctoring. IEEE Trans Multimedia 19(7):1609–1624. https://doi.org/10.1109/TMM.2017.2656064
3. Yaqub W, Mohanty M, Suleiman B (2021) Image-hashing-based anomaly detection for privacy-preserving online proctoring. arXiv:2107.09373
4. Learning M (2021) Proctoru. ProctorU website. Accessed on April 28, 2023
5. Feng X, Jiang Y, Yang X, Du M, Li X (2019) Computer vision algorithms and hardware implementations: a survey. Integration 69:309–320. https://doi.org/10.1016/j.vlsi.2019.07.005
6. Blagojević M, Milošević D (2015) Massive open online courses: Edx vs moodle mooc. In: Proc. 5th International Conference on Information Society and Technology, Kopaonik, Serbia, pp. 346–351

7. Baji T (2018) Evolution of the gpu device widely used in ai and massive parallel processing. In: 2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM), pp. 7–9. https://doi.org/10.1109/EDTM.2018.8421507
8. Browser SE (2021) Safe Exam Browser. https://safeexambrowser.org/. Accessed on April 28, 2023
9. Mercer (2021) Mettl. https://mettl.com/. Accessed on April 28, 2023
10. Technologies V (2021) ProctorTrack. https://www.proctortrack.com/. Accessed on April 28, 2023
11. Wahid A, Sengoku Y, Mambo M (20115) Toward constructing a secure online examination system. https://doi.org/10.1145/2701126.2701203
12. Li X, Chang Km, Yuan Y, Hauptmann A (2015) Massive open online proctor, pp. 1129–1137. https://doi.org/10.1145/2675133.2675245
13. Messerschmidt M, Pleva M (2019) Biometric systems utilizing neural networks in the authentication for e-learning platforms, pp. 518–523. https://doi.org/10.1109/ICETA48886.2019.9040132
14. Ganidisastra AHS, Bandung Y (2021) An incremental training on deep learning face recognition for m-learning online exam proctoring. In: 2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), pp. 213–219. https://doi.org/10.1109/APWiMob51111.2021.9435232
15. Ahmad I, Alqurashi F, Abozinadah E, Mehmood R (2021) A novel deep learning-based online proctoring system using face recognition, eye blinking, and object detection techniques. International Journal of Advanced Computer Science and Applications 12. https://doi.org/10.14569/IJACSA.2021.0121094
16. Zhang K, Zhang Z, Li Z, Qiao Y (2016) Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process Lett 23(10):1499–1503
17. Qi D, Tan W, Yao Q, Liu J (2023) Yolo5face: why reinventing a face detector. In: Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V, pp. 228–244. Springer
18. Deng J, Guo J, Ververas E, Kotsia I, Zafeiriou S (2020) Retinaface: single-shot multi-level face localisation in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5203–5212
19. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
20. Li X, Xu Y, Lv Q, Dou Y (2016) Affine-transformation parameters regression for face alignment. IEEE Signal Process Lett 23:55–59
21. Xia J, Zhang H, Wen S, Yang S, Xu M (2022) An efficient multitask neural network for face alignment, head pose estimation and face tracking. Expert Syst Appl 205:117368. https://doi.org/10.1016/j.eswa.2022.117368
22. Xia J, Qu W, Huang W, Zhang J, Wang X, Xu M (2022) Sparse local patch transformer for robust face alignment and landmarks inherent relation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4052–4061
23. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. https://doi.org/10.1109/CVPR.2016.90
24. Tan M, Le Q (2019) Efficient Net: rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research 97:6105–6114
25. Wang Q, Zhang P, Xiong H, Zhao J (2021) Face.evolve: a high-performance face recognition library. arXiv:2107.08621
26. Guo Y, Zhang L, Hu Y, He X, Gao J (2016) Ms-celeb-1m: a dataset and benchmark for large-scale face recognition. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III 14, pp. 87–102. Springer
27. Deng J, Guo J, Xue N, Zafeiriou S (2019) Arcface: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4690–4699
28. Wang A, Chen H, Liu L, Chen K, Lin Z, Han J, Ding G (2024) Yolov10: Real-time end-to-end object detection. arXiv:2405.14458
29. Ruiz N, Chong E, Rehg JM (2018) Fine-grained head pose estimation without keypoints. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2155–215509. https://doi.org/10.1109/CVPRW.2018.00281
30. Python (2020) Python Global Interpreter Lock. https://wiki.python.org/moin/GlobalInterpreterLock
31. Sommerlad P (2003) Reverse proxy patterns. In: EuroPLoP, pp. 431–458. Citeseer
32. Richardson L, Ruby S (2008) RESTful Web Services. O'Reilly Media, Inc
33. Quang PH (2024) VN-celeb: Vietnamese Celebrity Face Data and the Face Recognition Problem. https://viblo.asia/p/vn-celeb-du-lieu-khuon-mat-nguoi-noi-tieng-viet-nam-va-bai-toan-face-recognition-Az45bG9VKxY. Accessed: 2024-07-30

34. Tweissi A, Etaiwi W, Al-Eisawi D (2022) The accuracy of ai-based automatic proctoring in online exams. Electronic Journal of e-Learning 20. https://doi.org/10.34190/ejel.20.4.2600
35. Nurpeisova A, Shaushenova A, Mutalova Z, Ongarbayeva M, Niyazbekova S, Bekenova A, Zhumaliyeva L, Zhumasseitova S (2023) Research on the development of a proctoring system for conducting online exams in kazakhstan. Computation 11(6). https://doi.org/10.3390/computation11060120

Authors and Affiliations

Tuan Linh Dang[1] · Nguyen Minh Nhat Hoang[1] · The Vu Nguyen[1] · Hoang Vu Nguyen[1] · Quang Minh Dang[1] · Quang Hai Tran[1] · Huy Hoang Pham[1]

✉ Tuan Linh Dang
linhdt@soict.hust.edu.vn

Nguyen Minh Nhat Hoang
nhat.hnm194445@sis.hust.edu.vn

The Vu Nguyen
vu.nt194214@sis.hust.edu.vn

Hoang Vu Nguyen
vu.nh190100@sis.hust.edu.vn

Quang Minh Dang
minh.dq194796@sis.hust.edu.vn

Quang Hai Tran
hai.tq194755@sis.hust.edu.vn

Huy Hoang Pham
hoangph@soict.hust.edu.vn

[1] School of Information and Communications Technology, Hanoi University of Science and Technology, 01 Dai Co Viet Road, Hanoi 100000, Vietnam