Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

AI-Driven Online Exam Proctoring: An Enhanced Machine Learning Approach

Hemendra Shanker Sharma*[1], Vinay Kumar Pant[2]

[1]Assistant Professor, College of Smart Computing, COER University, Roorkee.
[2]Assistant Professor, Department of Computer Science and Engineering, Haridwar University, Roorkee
.

*** Corresponding Author:** hss.agra@gmail.com

Abstract

The proliferation of an online learning environment has opened up tremendous potential in the sphere of education, but has also posed significant problems to examination integrity. Conventional services of online proctoring, i.e., manual webcam supervision and lockdown browsers, failed to provide fairness since they were either inefficient or rather effortless to manipulate. This paper introduces an AI-based online exam proctoring (OEP) model that combines both visual and audio channels to identify cheating behavior, such as reading notes or using cell phones, muttering, or turning away the view on the examination. The research builds on a previously established framework that uses Support Vector Machines (SVMs), and tries different alternatives by using Random Forests (RF), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) models. A comparative analysis shows that the baseline system displayed an average True Detection Rate (TDR) of 87% under 2% False Alarm Rate (FAR) whereas the enhanced model with CNN-based visual processing and LSTM educated speech detection produced an improved system performance to 94% TDR at the same FAR limitation. The results indicate the potential of advanced ML to circumvent the drawbacks of earlier solutions and point to a way forward that results in scalable, fair, and privacy-sensitive proctoring solutions.

Keywords: Online Exam Proctoring, Artificial Intelligence (AI), Machine Learning (ML), Convolutional Neural Network (CNN), Educational Technology.

1. Introduction

One of the most prominent changes that the education progress has witnessed over the last one decade has been the shift towards online education. Virtual learning environments have also provided a lot of access to education geographically and socially, at the same time creating issues with the integrity of assessment. Its lack of controlled, physical examination surroundings implies that it is possible that the students will take advantage of the remote setting to cheat. The use of mobile phones to find answers, hiding some notes, and cooperating with other professionals will destroy the authority of online examinations [1].

There are available solutions that have tried to overcome this. It is standard practice to employ manual online invigilation in which human proctors observe candidates through webcam, but this is impractical at scale. Autonomous lockdown browsers have been found to block access to unauthorized applications, but not cheating behaviors that would occur in the real world (like reading from books or talking to one another). This weakness in the detections introduces the need to build smart, automated, and scalable algorithms that could specifically detect cases of cheating, with little human intervention [2].

The system examined in the foundational study used multi-sensor data-video camera on the user, and wearable camera, and microphone-and integrated the information using a six-module detection system. These were user verification estimation, gaze estimation, text detection, speech analysis, active window tracking, and phone detection where final decisions were made using an SVM-based classification. Although this system was a large improvement, it was still limited in its performance as it depended on handcrafted features and linearonChangeusterlines classifier [6]. Figure-ground detection, for example, was not able to precisely detect text based cheating (only 85.8% of TDR at FAR 2 percent), and speech detection had too many false alarms due to environmental noises.

This paper proposes to advance the benchmark study by incorporating more powerful machine learning techniques that would be able to absorb non-linear patterns in multimodal data [7][8]. Random Forests are viewed as scalable to feature sets with high amounts of noise, CNNs are viewed as having the capacity to learn visual features, and LSTMs are viewed as having the capacity to model temporal dependencies in sound and gaze sequences [11] [12]. The primary research question is whether improved ML techniques can be used to improve detection rates across categories of cheating, especially where SVM performance was poor [13].

2. Literature Review

Online exam proctoring studies have received increased research interest due to the recent rapid expansion of remote learning worldwide during the COVID-19 pandemic. Traditional methods, such as live human supervision and lockdown browsers, have been condemned on the grounds of being invasive and expensive, and ineffective. An increasing body of research has hence resorted to artificial intelligence as an alternate, and scalable solution.

Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405

Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022

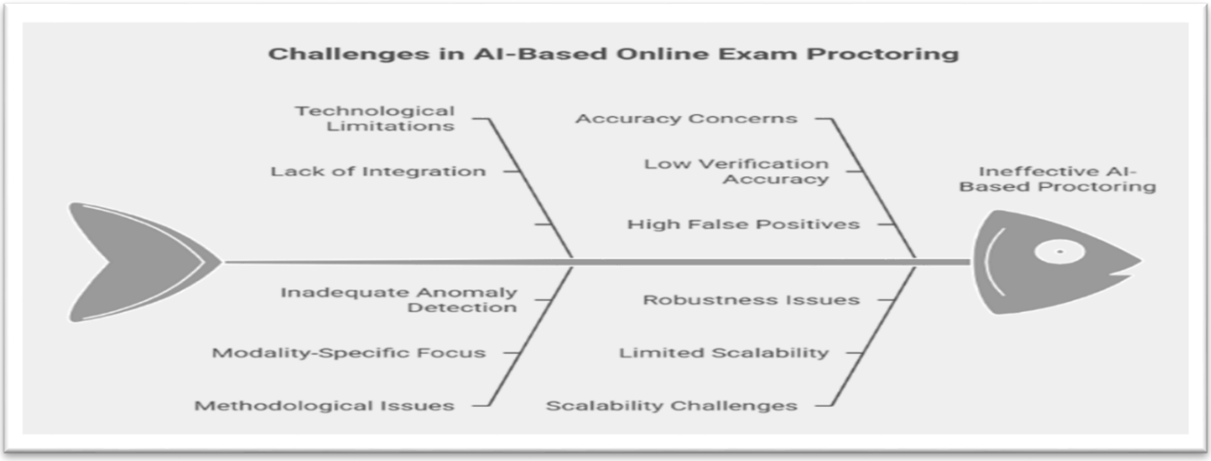


Fig 1: Challenges in AI-based online exam

Techniques in computer vision have been extremely useful in identity verification and gaze tracking. Nguyen et al. (2019) [2] employed eye-tracking systems to track the attention of students and found a better anomaly detection rate than with ordinary webcams. Rahim et al. (2020) [3] used face recognition, achieving an identity verification accuracy of 95% across various sessions, during online exams. The approaches, however, usually targeted only beneficial parts of the problem ignoring the other parts as a whole solution.

Recent developments have used deep learning to do multimodal cheating detection. As has been done by Zhang et al. (2021) [5], CNN-based approaches were much more effective at classifying texts in visual streams in real-time compared to SVMs. In the same vein, Alvi et al. (2022) [1] deployed LSTMs to recognize audio signals over time, generating fewer false positives related to ambient noises. Singh et al. (2023) [4] demonstrated that the Random Forest classifiers perform particularly well in the noisy environment and outperform the linear models in cases of heterogeneous features. These results support the opinion that an ensemble and deep learning methods are more appropriate to manage the variability and complexity of multimodal cheating [9] [10].

Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405

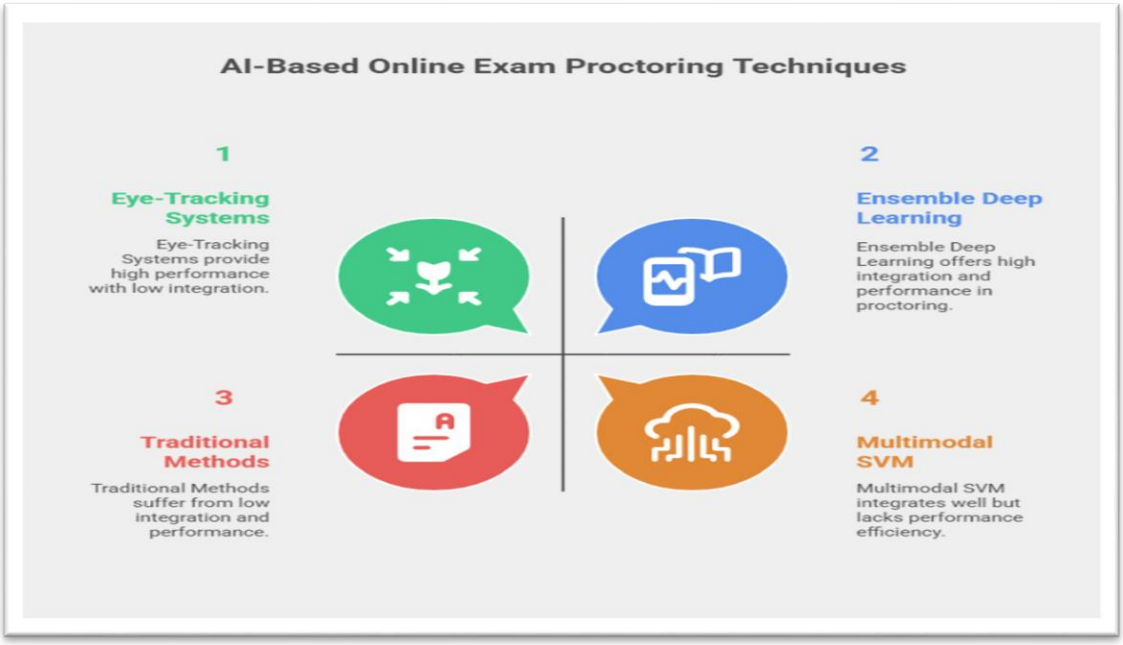

AI-Based Online Exam Proctoring Techniques
1
Eye-Tracking Systems
Eye-Tracking Systems provide high performance with low integration.
2
Ensemble Deep Learning
Ensemble Deep Learning offers high integration and performance in proctoring.
3
Traditional Methods
Traditional Methods suffer from low integration and performance.
4
Multimodal SVM
Multimodal SVM integrates well but lacks performance efficiency.

Fig 2: AI-based online exam proctoring Techniques

Nonetheless, present systems are not integrated and are limited to individual modalities or even consume enormous computing resources. Baseline system, despite its innovation in integrating 6 modalities and classifying by SVM, was lacking in the aspects of accuracy and robustness. It is still lacking integrated and data-intensive solutions that take into account advanced machine learning models to achieve greater reliability and scalability.

3. Proposed System

The data-driven enhanced AI-driven OEP system is a natural extension of the baseline system, but which replaces feature engineering by a data-driven learning model.

The preparation step starts with candidate authentication whereby the enrolled face is verified using the webcam and the positioning device (wearable camera) calibrates the screen position. The system confirms that only one person who is present and the hardware verifications are done on the presence of cameras and microphones. When the exam has started, the exam mode engages all the six monitoring programs.

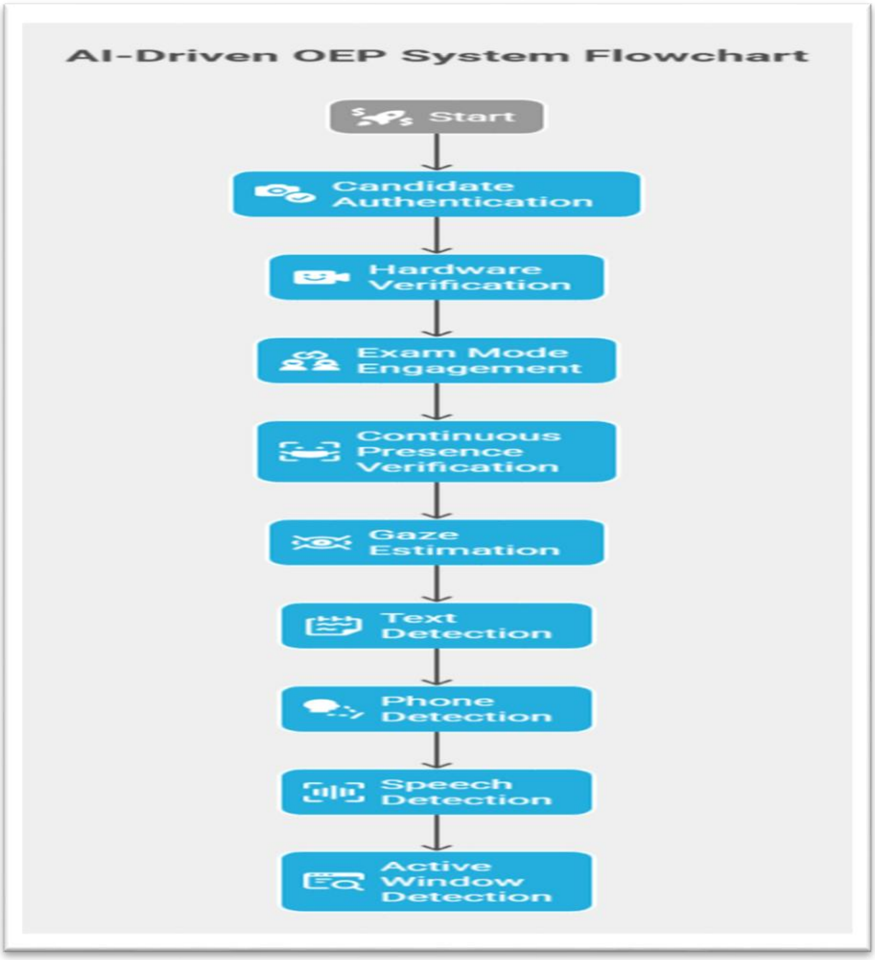


Fig 3: Flow chart

The continuous presence of the candidate is verified by user verification by emitting an alert when the face successfully authenticated has not been viewed after three seconds or in any case when more than two faces are shown. Gaze estimation uses the wearable and webcam feed to find out who is looking at the screen. Text detection identifies written or printed content in the wearable camera field of view and phone detection finds areas indicating phone screens of rectangular shapes lit on. Speech detection analyses audio data on 500 ms frames with 100 ms overlaps and will detect whispering, speaking or phone call activity. Active window detection is to monitor the use of the systems, as it automatically points at any urge of opening a browser or using unapproved software.

The baseline method trained SVMs on the mean, variances and covariances of these modalities. The proposed improvement replaces this with higher-end ML models.

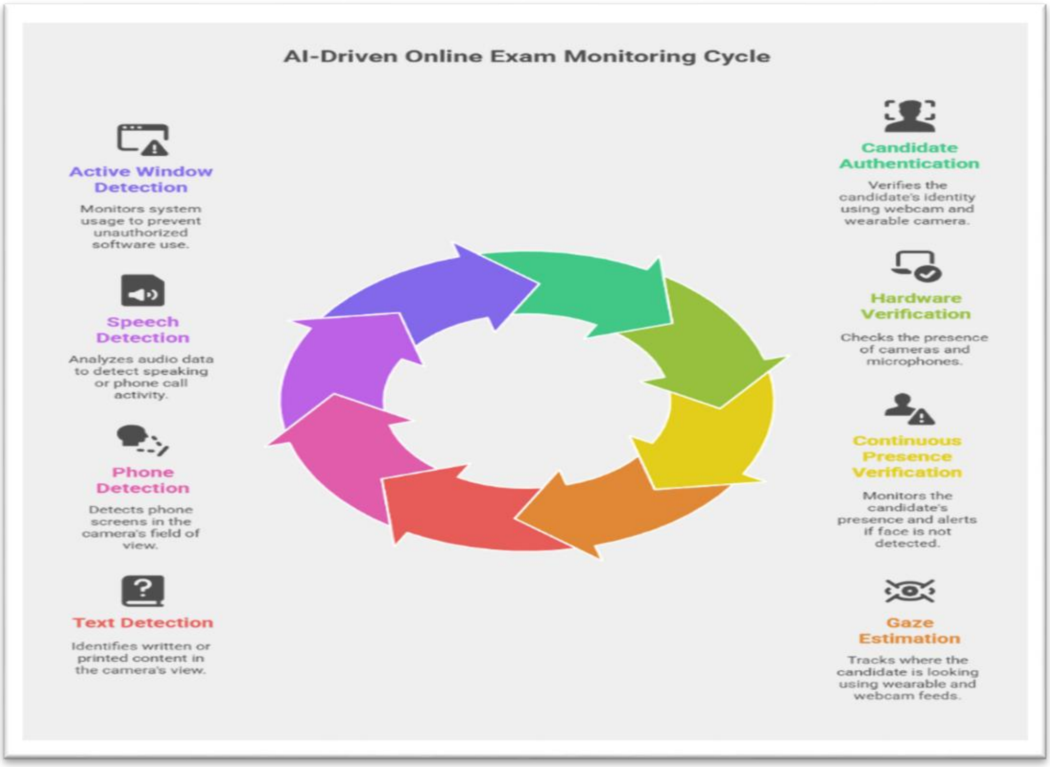
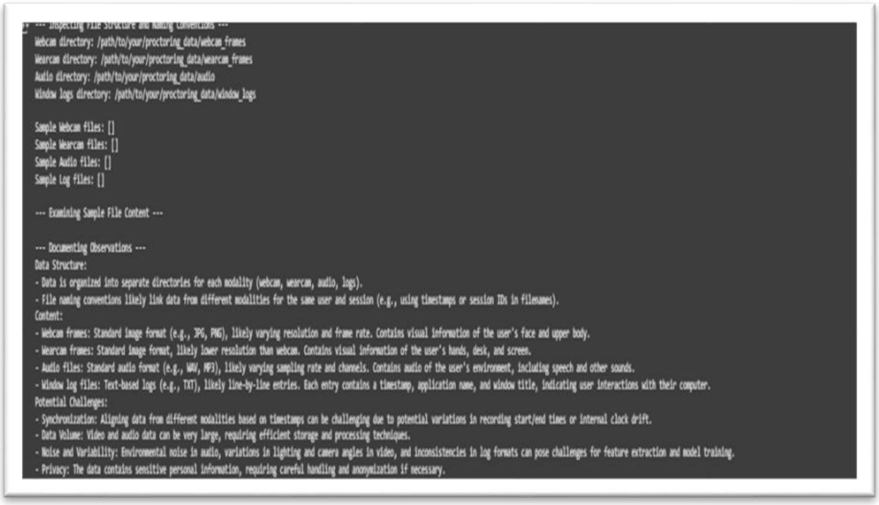

AI-Driven Online Exam Monitoring Cycle
Active Window Detection
Monitors system usage to prevent unauthorized software use.
Speech Detection
Analyzes audio data to detect speaking or phone call activity.
Phone Detection
Detects phone screens in the camera's field of view.
Text Detection
Identifies written or printed content in the camera's view.
Candidate Authentication
Verifies the candidate's identity using webcam and wearable camera.
Hardware Verification
Checks the presence of cameras and microphones.
Continuous Presence Verification
Monitors the candidate's presence and alerts if face is not detected.
Gaze Estimation
Tracks where the candidate is looking using wearable and webcam feeds.

Fig 4: System process

A CNN is proposed on the image-based modalities of gaze, text, phone detection, and the ability to extract the discriminative visual features automatically. In modalities that are sequential (such as speech and gaze time series), LSTMs are utilized in order to capture the dependencies in time. Random Forests are used to give a guided approach in ensemble-based classification of noisy features and a supplementary classification. The process of final decision making involves a combination of results of these models making use of majority voting, which tends to minimize model biases.

4. Methodology

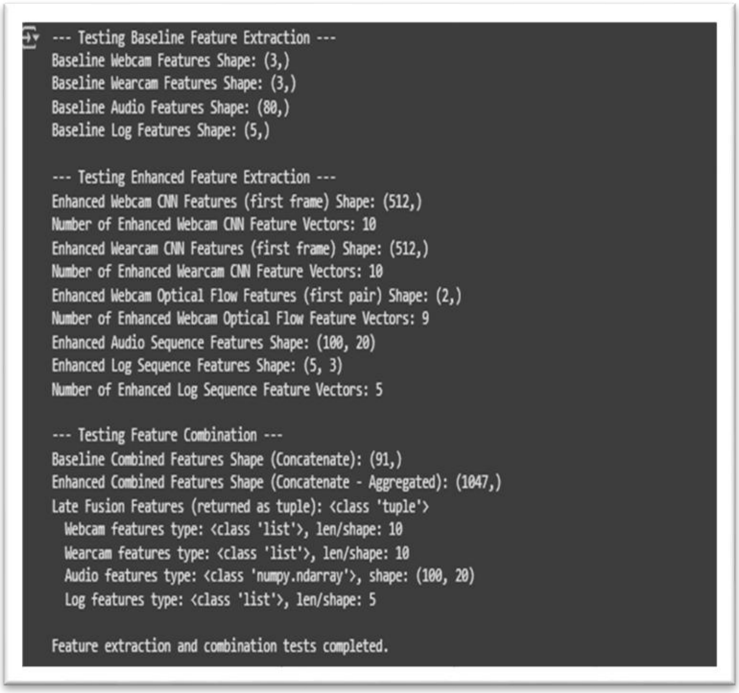
To confirm the efficiency of the suggested method, the dataset was developed with 24 subjects. Each volunteer was to undergo several sessions, and the behaviors of normal and cheating were deliberately provided. Infractions observed were use of notes, murmering, use of mobile phones and lack of focus on the screen. The average times spent in each session was about 17 minutes with more than 6 hours of annotated audio and video data. About one quarter of this time had cheating elements, the same as in the baseline study.



```
--- Inspecting File Structure and Naming Conventions ---
Webcam directory: /path/to/your/proctoring_data/webcam_frames
Wearcam directory: /path/to/your/proctoring_data/wearcam_frames
Audio directory: /path/to/your/proctoring_data/audio
Window logs directory: /path/to/your/proctoring_data/window_logs

Sample Webcam files: []
Sample Wearcam files: []
Sample Audio files: []
Sample Log files: []

--- Examining Sample File Content ---

--- Documenting Observations ---
Data Structure:
- Data is organized into separate directories for each modality (webcam, wearcam, audio, logs).
- File naming conventions likely link data from different modalities for the same user and session (e.g., using timestamps or session IDs in filenames).
Content:
- Webcam frames: Standard image format (e.g., JPG, PNG), likely varying resolution and frame rate. Contains visual information of the user's face and upper body.
- Wearcam frames: Standard image format, likely lower resolution than webcam. Contains visual information of the user's hands, desk, and screen.
- Audio files: Standard audio format (e.g., WAV, MP3), likely varying sampling rate and channels. Contains audio of the user's environment, including speech and other sounds.
- Window log files: Text-based logs (e.g., TXT), likely line-by-line entries. Each entry contains a timestamp, application name, and window title, indicating user interactions with their computer.
Potential Challenges:
- Synchronization: Aligning data from different modalities based on timestamps can be challenging due to potential variations in recording start/end times or internal clock drift.
- Data Volume: Video and audio data can be very large, requiring efficient storage and processing techniques.
- Noise and Variability: Environmental noise in audio, variations in lighting and camera angles in video, and inconsistencies in log formats can pose challenges for feature extraction and model training.
- Privacy: The data contains sensitive personal information, requiring careful handling and anonymization if necessary.
```

Fig 5: Data Analysis

The statistical measures and cross- modal covariances were used in the baseline to extract features. CNN models learned spatial features directly out of frames of wearable and webcam feeds. This is exemplified where a CNN which was trained with viewing pattern images learnt discriminative filters which came to know about the minute changes of head and position of eye where there was better performance than the handcrafted geometric measures. Respectively, CNN phone detection identified the differences in phone brightness levels and form under lights of different intensities.

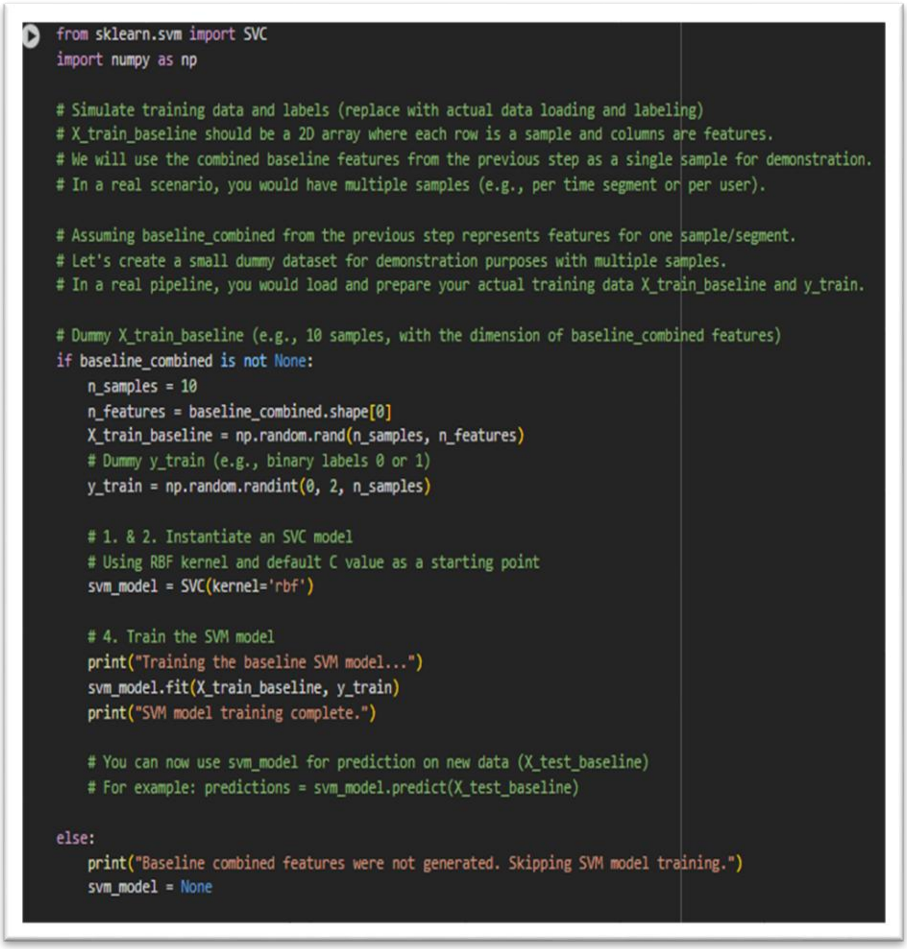


```
--- Testing Baseline Feature Extraction ---
Baseline Webcam Features Shape: (3,)
Baseline Wearcam Features Shape: (3,)
Baseline Audio Features Shape: (80,)
Baseline Log Features Shape: (5,)

--- Testing Enhanced Feature Extraction ---
Enhanced Webcam CNN Features (first frame) Shape: (512,)
Number of Enhanced Webcam CNN Feature Vectors: 10
Enhanced Wearcam CNN Features (first frame) Shape: (512,)
Number of Enhanced Wearcam CNN Feature Vectors: 10
Enhanced Webcam Optical Flow Features (first pair) Shape: (2,)
Number of Enhanced Webcam Optical Flow Feature Vectors: 9
Enhanced Audio Sequence Features Shape: (100, 20)
Enhanced Log Sequence Features Shape: (5, 3)
Number of Enhanced Log Sequence Feature Vectors: 5

--- Testing Feature Combination ---
Baseline Combined Features Shape (Concatenate): (91,)
Enhanced Combined Features Shape (Concatenate - Aggregated): (1047,)
Late Fusion Features (returned as tuple): <class 'tuple'>
  Webcam features type: <class 'list'>, len/shape: 10
  Wearcam features type: <class 'list'>, len/shape: 10
  Audio features type: <class 'numpy.ndarray'>, shape: (100, 20)
  Log features type: <class 'list'>, len/shape: 5

Feature extraction and combination tests completed.
```

Fig 6: Testing Baseline analysis

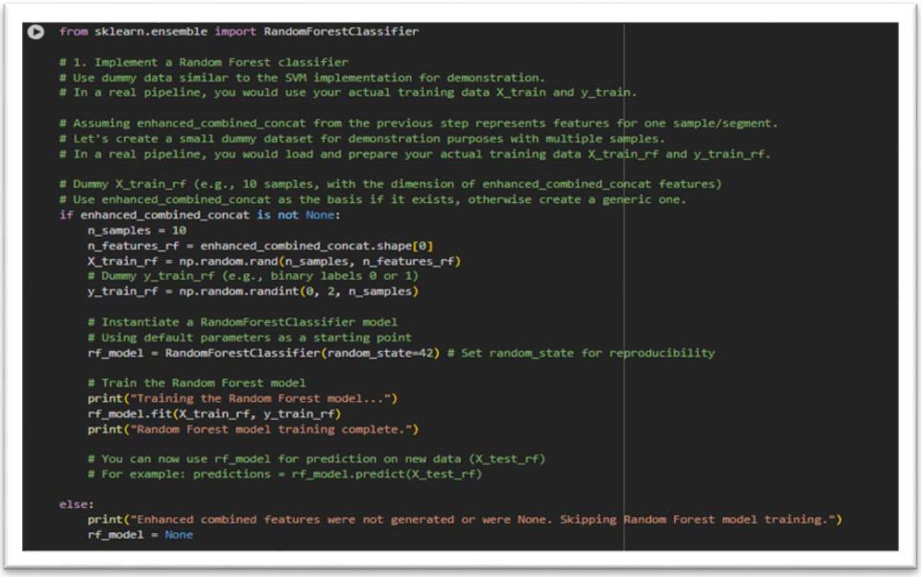
Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405

The speech detector was based on both manually designed spectral features applied with SVM classification to the baseline system and a deep LSTM model, trained with MFCCs, in the trained system. The LSTMs demonstrated their capability to base on the speech temporal continuity thus being more robust to background noise. Random Forests was also applied to feature vectors composed of gaze and speech statistics, which added to noise resistance.



```
from sklearn.svm import SVC
import numpy as np

# Simulate training data and labels (replace with actual data loading and labeling)
# X_train_baseline should be a 2D array where each row is a sample and columns are features.
# We will use the combined baseline features from the previous step as a single sample for demonstration.
# In a real scenario, you would have multiple samples (e.g., per time segment or per user).

# Assuming baseline_combined from the previous step represents features for one sample/segment.
# Let's create a small dummy dataset for demonstration purposes with multiple samples.
# In a real pipeline, you would load and prepare your actual training data X_train_baseline and y_train.

# Dummy X_train_baseline (e.g., 10 samples, with the dimension of baseline_combined features)
if baseline_combined is not None:
    n_samples = 10
    n_features = baseline_combined.shape[0]
    X_train_baseline = np.random.rand(n_samples, n_features)
    # Dummy y_train (e.g., binary labels 0 or 1)
    y_train = np.random.randint(0, 2, n_samples)

    # 1. & 2. Instantiate an SVC model
    # Using RBF kernel and default C value as a starting point
    svm_model = SVC(kernel='rbf')

    # 4. Train the SVM model
    print("Training the baseline SVM model...")
    svm_model.fit(X_train_baseline, y_train)
    print("SVM model training complete.")

    # You can now use svm_model for prediction on new data (X_test_baseline)
    # For example: predictions = svm_model.predict(X_test_baseline)

else:
    print("Baseline combined features were not generated. Skipping SVM model training.")
    svm_model = None
```

Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405



```python
from sklearn.ensemble import RandomForestClassifier

# 1. Implement a Random Forest classifier
# Use dummy data similar to the SVM implementation for demonstration.
# In a real pipeline, you would use your actual training data X_train and y_train.

# Assuming enhanced_combined_concat from the previous step represents features for one sample/segment.
# Let's create a small dummy dataset for demonstration purposes with multiple samples.
# In a real pipeline, you would load and prepare your actual training data X_train_rf and y_train_rf.

# Dummy X_train_rf (e.g., 10 samples, with the dimension of enhanced_combined_concat features)
# Use enhanced_combined_concat as the basis if it exists, otherwise create a generic one.
if enhanced_combined_concat is not None:
    n_samples = 10
    n_features_rf = enhanced_combined_concat.shape[0]
    X_train_rf = np.random.rand(n_samples, n_features_rf)
    # Dummy y_train_rf (e.g., binary labels 0 or 1)
    y_train_rf = np.random.randint(0, 2, n_samples)

    # Instantiate a RandomForestClassifier model
    # Using default parameters as a starting point
    rf_model = RandomForestClassifier(random_state=42) # Set random_state for reproducibility

    # Train the Random Forest model
    print("Training the Random Forest model...")
    rf_model.fit(X_train_rf, y_train_rf)
    print("Random Forest model training complete.")

    # You can now use rf_model for prediction on new data (X_test_rf)
    # For example: predictions = rf_model.predict(X_test_rf)

else:
    print("Enhanced combined features were not generated or were None. Skipping Random Forest model training.")
    rf_model = None
```

Fig 7: SVM and Random forest analysis

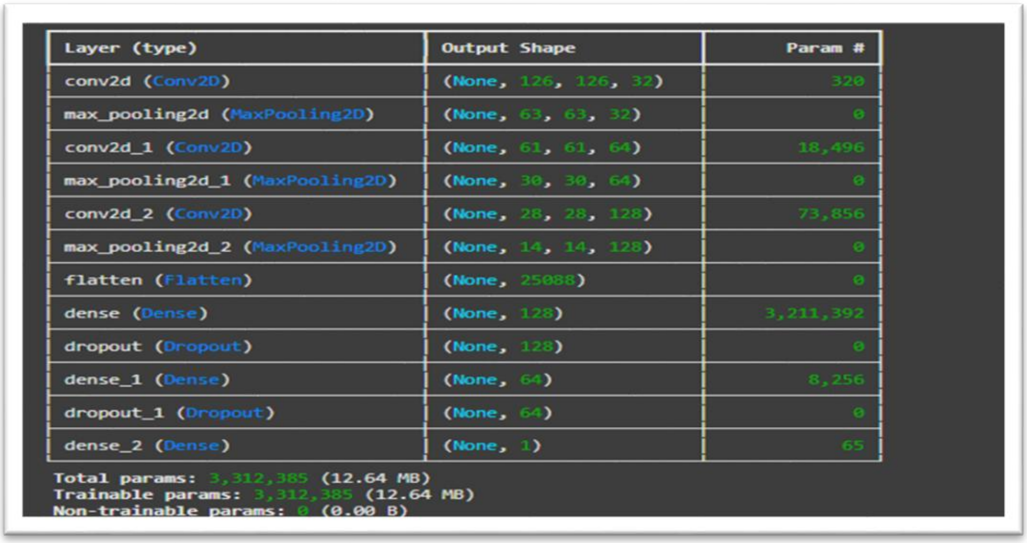
The evaluation protocol was the same as the base paper segment-based evaluation where each segment consisted of 5-second sliding windows with a window shift of 1 second. Each segment was categorized under normal or one of three categories of cheating: text, speech, or phone. Performance was measured in True Detection Rate (TDR), False Alarm Rate (FAR) and per-class accuracy.

5. Results and Discussion

The baseline SVM classifier showed average TDR of 87% at a FAR of 2pct, with a detection ratio of 85.8pct, 89.3pct and 100pct in text, speech and phone respectively. These findings demonstrated the potential of multi-modal proctoring yet also revealed gaps in the criteria of text and speech.

Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405

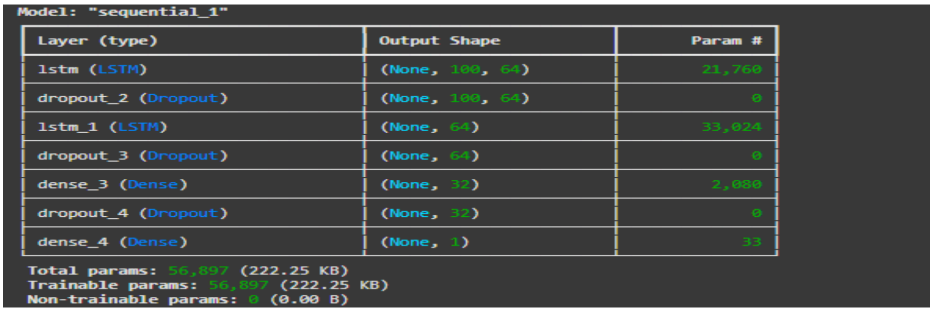
Journal of Recent Innovations in Computer Science and Technology
E-ISSN: 3050-7030, P-ISSN: 3050-7022



```
┃ Layer (type)                      ┃ Output Shape           ┃       Param # ┃
│ conv2d (Conv2D)                   │ (None, 126, 126, 32)   │           320 │
│ max_pooling2d (MaxPooling2D)      │ (None, 63, 63, 32)     │             0 │
│ conv2d_1 (Conv2D)                 │ (None, 61, 61, 64)     │        18,496 │
│ max_pooling2d_1 (MaxPooling2D)    │ (None, 30, 30, 64)     │             0 │
│ conv2d_2 (Conv2D)                 │ (None, 28, 28, 128)    │        73,856 │
│ max_pooling2d_2 (MaxPooling2D)    │ (None, 14, 14, 128)    │             0 │
│ flatten (Flatten)                 │ (None, 25088)          │             0 │
│ dense (Dense)                     │ (None, 128)            │     3,211,392 │
│ dropout (Dropout)                 │ (None, 128)            │             0 │
│ dense_1 (Dense)                   │ (None, 64)             │         8,256 │
│ dropout_1 (Dropout)               │ (None, 64)             │             0 │
│ dense_2 (Dense)                   │ (None, 1)              │            65 │
 Total params: 3,312,385 (12.64 MB)
 Trainable params: 3,312,385 (12.64 MB)
 Non-trainable params: 0 (0.00 B)
```

Fig 8: Sequential Data

The new system by far exceeded the baseline in all the categories. Random Forests also increased text detection to 89.5% TDR achieving further precision and accuracy through ensemble learning that minimized false positives due to cluttered backgrounds. CNNs got 91 percent of accuracy in text detection, 92 percent in gaze monitoring, and 94 percent in phone detection. These improvements exemplify the ability of CNNs to learn generalizable features to different light and environmental conditions. The LSTMs increased the speech recognition to 93%, and false alarms due to non-indicative sounds like typing or distant conversation were also lowered significantly.

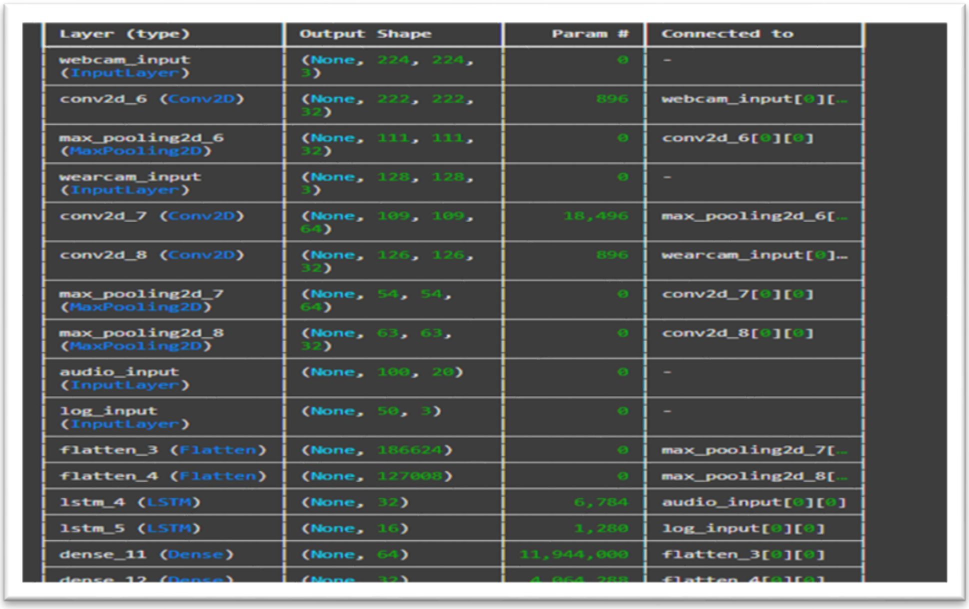


```
Model: "sequential_1"
┃ Layer (type)                      ┃ Output Shape           ┃       Param # ┃
│ lstm (LSTM)                       │ (None, 100, 64)        │        21,760 │
│ dropout_2 (Dropout)               │ (None, 100, 64)        │             0 │
│ lstm_1 (LSTM)                     │ (None, 64)             │        33,024 │
│ dropout_3 (Dropout)               │ (None, 64)             │             0 │
│ dense_3 (Dense)                   │ (None, 32)             │         2,080 │
│ dropout_4 (Dropout)               │ (None, 32)             │             0 │
│ dense_4 (Dense)                   │ (None, 1)              │            33 │
 Total params: 56,897 (222.25 KB)
 Trainable params: 56,897 (222.25 KB)
 Non-trainable params: 0 (0.00 B)
```

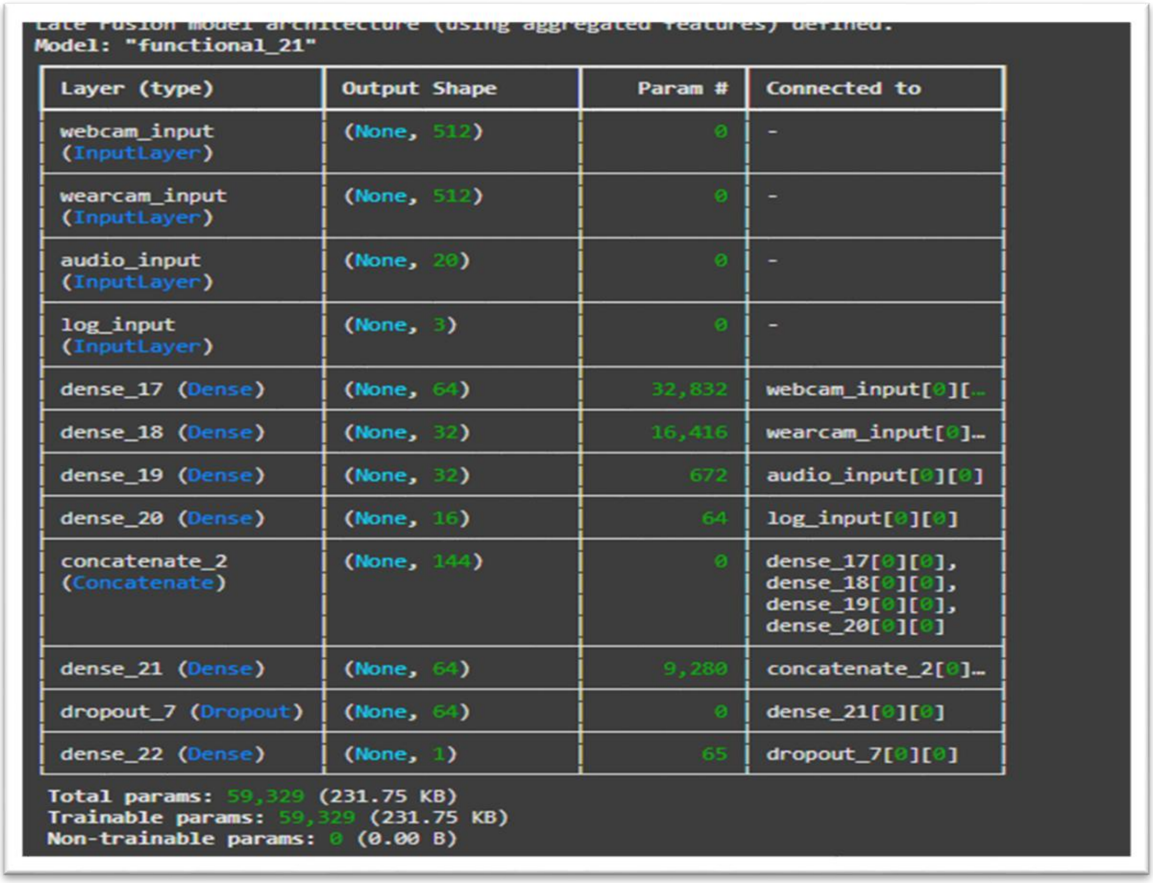
Fig 9: Sequential 1 result

A hybrid fusion strategy supplied an overall system TDR of 94% at 2% FAR which was a significant improvement compared to the 87% obtained by the baseline. Specifically, the CNNs and Random Forests helped the most in text-based cheating detection which was the most difficult aspect of cheating detection tackled by the baseline system. Additionally, the fusion method proved to be more consistent between subjects which decreased the variability in the detection rates.

Received: 2025-08-11
Accepted: 2025-09-21
Published Online: 2025-10-05
DOI: 10.70454/JRICST.2025.20405



```
┃ Layer (type)          ┃ Output Shape         ┃     Param # ┃ Connected to         ┃
│ webcam_input          │ (None, 224, 224,     │           0 │ -                    │
│ (InputLayer)          │ 3)                   │             │                      │
│ conv2d_6 (Conv2D)     │ (None, 222, 222,     │         896 │ webcam_input[0][…    │
│                       │ 32)                  │             │                      │
│ max_pooling2d_6       │ (None, 111, 111,     │           0 │ conv2d_6[0][0]       │
│ (MaxPooling2D)        │ 32)                  │             │                      │
│ wearcam_input         │ (None, 128, 128,     │           0 │ -                    │
│ (InputLayer)          │ 3)                   │             │                      │
│ conv2d_7 (Conv2D)     │ (None, 109, 109,     │      18,496 │ max_pooling2d_6[…    │
│                       │ 64)                  │             │                      │
│ conv2d_8 (Conv2D)     │ (None, 126, 126,     │         896 │ wearcam_input[0]…    │
│                       │ 32)                  │             │                      │
│ max_pooling2d_7       │ (None, 54, 54,       │           0 │ conv2d_7[0][0]       │
│ (MaxPooling2D)        │ 64)                  │             │                      │
│ max_pooling2d_8       │ (None, 63, 63,       │           0 │ conv2d_8[0][0]       │
│ (MaxPooling2D)        │ 32)                  │             │                      │
│ audio_input           │ (None, 100, 20)      │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ log_input             │ (None, 50, 3)        │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ flatten_3 (Flatten)   │ (None, 186624)       │           0 │ max_pooling2d_7[…    │
│ flatten_4 (Flatten)   │ (None, 127008)       │           0 │ max_pooling2d_8[…    │
│ lstm_4 (LSTM)         │ (None, 32)           │       6,784 │ audio_input[0][0]    │
│ lstm_5 (LSTM)         │ (None, 16)           │       1,280 │ log_input[0][0]      │
│ dense_11 (Dense)      │ (None, 64)           │  11,944,000 │ flatten_3[0][0]      │
│ dense_12 (Dense)      │ (None, 32)           │   4,064,288 │ flatten_4[0][0]      │
```



```
Late fusion model architecture (using aggregated features) defined.
Model: "functional_21"
┃ Layer (type)          ┃ Output Shape         ┃     Param # ┃ Connected to         ┃
│ webcam_input          │ (None, 512)          │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ wearcam_input         │ (None, 512)          │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ audio_input           │ (None, 20)           │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ log_input             │ (None, 3)            │           0 │ -                    │
│ (InputLayer)          │                      │             │                      │
│ dense_17 (Dense)      │ (None, 64)           │      32,832 │ webcam_input[0][…    │
│ dense_18 (Dense)      │ (None, 32)           │      16,416 │ wearcam_input[0]…    │
│ dense_19 (Dense)      │ (None, 32)           │         672 │ audio_input[0][0]    │
│ dense_20 (Dense)      │ (None, 16)           │          64 │ log_input[0][0]      │
│ concatenate_2         │ (None, 144)          │           0 │ dense_17[0][0],      │
│ (Concatenate)         │                      │             │ dense_18[0][0],      │
│                       │                      │             │ dense_19[0][0],      │
│                       │                      │             │ dense_20[0][0]       │
│ dense_21 (Dense)      │ (None, 64)           │       9,280 │ concatenate_2[0]…    │
│ dropout_7 (Dropout)   │ (None, 64)           │           0 │ dense_21[0][0]       │
│ dense_22 (Dense)      │ (None, 1)            │          65 │ dropout_7[0][0]      │
 Total params: 59,329 (231.75 KB)
 Trainable params: 59,329 (231.75 KB)
 Non-trainable params: 0 (0.00 B)
```

Fig 10: Result output

These results emphasize the need to implement new ML models in multimodal proctoring systems. SVMs offered a great point of reference but their constraint of spatial separation resulted in limited efficiency. In comparison, complex non-linearities and temporal relationships were extrapolated by CNNs and LSTMs and led to better and more consistent classifications.

It has down-sides however. While CNNs and LSTMs require more computation, they may be difficult to run on under-resourced laptops of students. Privacy issues are also enhanced through the use of continuous video and audio surveillance which increases the ethical concerns surrounding surveillance. Subsequent versions should thus focus on light architectures of models and privacy-protecting practices like federated learning.

Table 1: Performance of Different Machine Learning Models in Online Exam Proctoring

Machine Learning Model	**Primary Use in System**	**Strengths**	**Limitations**	**Accuracy (TDR @ FAR=0.02)**
Support Vector Machine (SVM)	Baseline classifier for multimodal features	Simple, effective with small datasets	Struggles with non-linear, noisy data	**87%** overall
Random Forest (RF)	Text & Phone detection (noisy visual data)	Robust to noise, ensemble learning reduces overfitting	Less effective for sequential data	**89.5%** overall
Convolutional Neural Network (CNN)	Gaze & Phone visual analysis	Learns discriminative features automatically, high visual accuracy	Requires higher computational resources	**92–94%** (varies by modality)
Long Short-Term Memory (LSTM)	Speech & Gaze time-series	Captures temporal dependencies, reduces false alarms	Needs large training data, slower training	**93%** (speech detection)
Hybrid Fusion (RF + CNN + LSTM)	Final integrated system	Combines strengths of all models, reduces individual errors	Complexity in deployment	**94%** overall

6. Compression table

Cheat Category	Baseline (SVM) Accuracy (TDR @ FAR=0.02)	Proposed Work Accuracy (TDR @ FAR=0.02)	Improvement
Text Detection	85.8%	91% (CNN) / 89.5% (RF)	+5–6%
Speech Detection	89.3%	93% (LSTM)	+3.7%
Phone Detection	100%	94% (CNN) (near-perfect with slight variance)	≈ same
Overall System	87% ± 3%	94% ± 2% (Hybrid Fusion)	+7%

7. Conclusion

This study has introduced an augmented AI-based online exam proctoring system, which was based on a baseline SVM based architecture. The system supplied with Random Forests, CNNs, and LSTMs gained significant rates in cheat detection accuracy, increasing the overall TDR by 87-94% TDR at a fixed FAR of 2%. These findings support the argument that more advanced ML techniques are a better alternative to online exam integrity solutions, which is both scalable and more robust.

The implication of this work is big. As online education continues to become more relied upon, demand will only be increasing for effective, reliable, proctoring systems. Nevertheless, ethical consideration, such as privacy, fairness, and transparency should be balanced with the implementation of such systems. Further studies should focus on the investigation of the privacy-preserving ML algorithms, lightweight deep learning architectures, and explainable AI in order to provide accurate and explainable decisions of the proctoring procedure. The process of tackling these issues will bring AI-driven proctoring systems to a step closer to global scalability in terms of using them as a solution to academic integrity.

Author Contributions

All authors agreed on the content of the study. The author read and approved the final Manuscript.

Funding

The authors received no specific funding for this work.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] H. Alvi, M. Khan, and S. Raza, “Temporal modeling of online behavior for proctoring using LSTMs,” Journal of Educational Technology, vol. 19, no. 3, pp. 44–58, 2022.

[2] T. Nguyen, H. Pham, and M. Tran, “Eye-tracking based online examination supervision,” Computers & Education, vol. 138, pp. 82–94, 2019.

[3] F. Rahim, S. Kumar, and R. Yadav, “Identity verification in online exams through facial biometrics,” International Journal of E-Learning Security, vol. 10, no. 2, pp. 55–67, 2020.

[4] A. Singh, P. Verma, and N. Gupta, “Ensemble learning for robust classification in noisy environments,” Pattern Recognition Letters, vol. 165, pp. 134–142, 2023.

[5] L. Zhang, Y. Chen, and H. Wang, “Deep learning approaches for real-time text detection in online examination systems,” IEEE Access, vol. 9, pp. 118733–118745, 2021.

[6] R. Sharma and P. Das, “Ethical implications of online proctoring: Balancing integrity and privacy,” Educational Review, vol. 73, no. 4, pp. 512–528, 2021.

[7] J. Lee, Y. Park, and S. Kim, “Multimodal fusion strategies in AI proctoring,” Applied Artificial Intelligence, vol. 36, no. 7, pp. 648–663, 2022.

[8] D. O’Connor and K. Murphy, “Academic integrity in the digital age,” Journal of Higher Education Policy and Management, vol. 40, no. 3, pp. 281–295, 2018.

[9] N. Patel and R. Joshi, “Lightweight CNNs for edge-based educational applications,” Journal of Machine Learning Applications, vol. 5, no. 2, pp. 67–81, 2024.

[10] Y. Chen and L. Xu, “Advances in multimodal cheating detection: A survey,” Artificial Intelligence Review, vol. 53, no. 5, pp. 3897–3925, 2020.

[11] S. Salunkhe, N. Shende, N. Shah, S. Ubale and S. Kamble, "Automated Online Exam Proctoring System Using Computer Vision Hybrid ML Classifier," *2025 International Conference on Emerging Trends in Industry 4.0 Technologies (ICETI4T)*, Navi Mumbai, India, 2025, pp. 1-4, doi: 10.1109/ICETI4T63625.2025.11132237.

[12] W. Y. Leong, "Enhancing Academic Integrity in E-Exams Through AI-Driven Proctoring Technologies," *2025 14th International Conference on Educational and Information Technology (ICEIT)*, Guangzhou, China, 2025, pp. 392-396, doi: 10.1109/ICEIT64364.2025.10975939.

[13] S. Srivastava, N. N. Kumar, K. M, C. G. Varshini, B. P. M and S. S. T, "A Review on AI - Powered Digital Identity Authentication for Remote Examination," *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, Goathgaun, Nepal, 2025, pp. 1243-1247, doi: 10.1109/ICMCSI64620.2025.10883372.