Multi-Model Comparative Framework for Video Engagement Estimation using CNN-ViT Hybrid Architectures

Technical Report

By:

Abdalrahman Alshamaileh 165285

Nidal Shahin 162278

Abdalrahman Sheyab 164372

Mohammed khasawneh 161024

Course: Computer Vision course – Fall 2025/2026

Course instructor: Dr.Abdullah Al-amaren

Contents

1 Problem statement

Nowadays, students' engagement and interaction during a lesson are a core part of a successful learning process. Therefore, monitoring students during the lecture to estimate their engagement level is important for improving the learning process and ensuring a high quality of education.

Our aim is to build an **Engagement Monitoring System** that detects students' engagement level during class. This system helps the instructor evaluate each student's attention and engagement during the lecture. Consequently, the instructor can better understand each student's overall performance. Additionally, identifying the types of behavior that lead to lack of engagement can help the instructor adjust their approach and teaching style to stimulate students' attention during instruction.

The system detects students' faces and draws a *special* bounding box with a label. The term "special" refers to the bounding box color: **green** for an engaged student and **red** otherwise (and **orange** in some cases).

2 Research on the Neural Networks and architectures

2.1 Neural Networks used for the problem

Most (or nearly all) networks used to solve this problem are **CNN-based** models. For our dataset, multiple architectures were used, and each architecture follows a different approach for learning engagement-related patterns from facial cues and video clips.

2.2 Modern architectures

Below is a list of architectures commonly used for this task, where each has a specific goal:

1. **CNN-Only Architecture** (e.g., ResNet, VGG, MobileNetV3): these models treat the task as **image classification** by assigning the clip's engagement label to every extracted frame (as done in preprocessing). They are effective in facial feature extraction.

2. **CNN + RNN Architecture** (e.g., ResNet-50 + LSTM, VGG-16 + LSTM): the CNN extracts **spatial** features while the RNN (LSTM) learns **temporal dynamics**. Many papers using the same dataset report CNN+RNN as one of the best-performing approaches.

3. **Transformer-based Models** (e.g., TimeSformer, Video Swin Transformer, ViViT): these models split videos into **spatiotemporal patches** and model long-range dependencies using attention. Despite strong performance, they are heavy and complex.

2.3 Modern architectures comparison

For each category, we list a well-known representative model.

Table 1: Modern architectures used to solve attention/engagement monitoring.

Architecture	**Description and number/types of layers**	**Advantages**	**Disadvantages**
(CNN-Only) ResNet-34	34 convolutional layers with residual skip connections. Each frame is processed independently and classified using fully connected layers.	Simple implementation, fast training, strong spatial feature extraction for facial features.	Ignores temporal behavior; cannot model engagement changes over time; sensitive to noisy frame labels.
(CNN + RNN) ResNet-50 + LSTM	ResNet-50 extracts 2048-D features per frame; LSTM layers model the temporal sequence; followed by dense classification layers.	Strong temporal modeling; widely used for DAiSEE; effective for engagement patterns such as attention drift or fatigue.	Higher computational cost; longer training time (more than 7 hours on Kaggle GPU T4×2); more complex architecture.
(Transformer-based Video Model) TimeSformer	Divides video into spatiotemporal patches processed by multi-head self-attention layers and feed-forward blocks.	State-of-the-art performance; models long-range temporal dependencies; robust to noise.	Extremely computationally expensive; more complex implementation than CNN+RNN.

3 Models' development and training

3.1 Dataset: DAiSEE

DAiSEE (Dataset for Affective States in E-Learning Environments) obtained from Kaggle. A multi-label video dataset designed for recognizing user affective states in "in-the-wild" settings of e-learning. It contains **9000 video snippets** of 10 seconds each. Each clip is manually labeled for four levels of intensity (0: Very Low, 1: Low, 2: High, 3: Very

High) for the following states: **Boredom, Engagement, Confusion,** and **Frustration**. Videos were captured at **640 × 480** pixels with a frame rate of **30 fps**. The dataset is originally split into **Train, Validation, and Test** splits.

Preprocessing

The preprocessing steps were as follows:

1. We used **OpenCV (cv2)** to extract frames from each clip. The number of frames extracted was adjusted from **7 to 60 FPS** depending on the model used and the class label distribution.

2. **YuNet** was used to extract **12 facial landmarks**, where 4 of them define the face bounding box.

3. We used **OpenCV (cv2)** again to crop faces from frames.

4. Cropped face images were resized to **224 × 224** and fed into each model using the bounding boxes YuNet provided.

3.2 Training and validation

The training pipeline included various configurations. For optimizers, we tried two variants of Adam: **Adam** and **AdamW**. For loss functions, Multiple ones were evaluated, like **Binary Cross-Entropy**, **Cross-Entropy**, **Focal Loss**, **Mean Squared Error (MSE)**, and **CORAL Loss**.

Multiple model frameworks and configurations were evaluated in an extensive search for the optimal model that could counter the severe imbalance nature of the data set in addition to its small size, from simple classification and regression heads moving to attention and temporal ones with multiple label-based specialist heads and finally adversarial GANs-like discriminator and Gradiant Reversal Layer in attempt to prevent the models from collapsing to the majority classes.

Multiple batch sizes were explored (**4, 8, 16, and 64**). Larger batch sizes were assigned to smaller models to avoid **out-of-memory (OOM)** issues, which did occur when we tried 128 and 256 as batch sizes.

Several training techniques were applied:

- **Early stopping** with patience values ranging from **3 to 7**.

- **Data Parallelism**, using PyTorch's `DataParallel` to distribute computation across available GPUs, since the training was very exhaustive and computationally heavy.

Table 2: Description of the hyperparameters considered for each architecture.

Architecture	**Hyperparameter**	**Description**	**Value(s)**
MobileNetV2 + LSTM	Hidden Dimension, Weight Decay, LR, Unfreeze Percentage, Smoothness Loss Weight, Consistency Loss Weight	**Hidden Dimension**: size of LSTM hidden state controlling temporal capacity. **LR**: step size for parameter updates. **Weight Decay**: regularization to reduce overfitting. **Unfreeze Percentage**: fraction of pretrained backbone layers fine-tuned. **Smoothness Loss Weight**: enforces smooth temporal predictions. **Consistency Loss Weight**: enforces prediction consistency across time.	Hidden Dim: 128, 256, 512 LR: $1e^{-5}$ to $5e^{-4}$ Weight Decay: $1e^{-6}$, $1e^{-3}$ Unfreeze: 0, 25%, 50%, 75% Smoothness: 0–0.5 Consistency: 0–0.5
ResNet34 + LSTM	Same as above.	Same as above.	Hidden Dim: 256, 512 LR: $5e^{-5}$ to $2e^{-4}$ Weight Decay: $1e^{-5}$, $5e^{-4}$ Unfreeze: 0, 25%, 50%, 75% Smoothness: 0–0.5 Consistency: 0–0.5
ViT + GRL + Discriminator	Learning Rate, GRL Adversarial Weight	**LR**: controls execution speed of transformer updates. **Adversarial Weight**: Determines the influence the discriminator has on the backbone.	LR: $1e^{-4}$, $3e^{-4}$, $5e^{-4}$ Adversarial Weight: 5%, 10%, 20%

Table 3: Combinations of hyperparameter values and corresponding performance.

Architecture	**Hyperparameter combination**	**Training performance**	**Validation performance**
MobileNetV2 + LSTM	Hidden Dim: 128 LR: 0.00002 Weight Decay: 0.00033 Smoothness Loss: 0.12151 Consistency Loss: 0.00427 Unfreeze: 50%	Loss: 0.3615 RMSE: 0.7174 Kappa: -0.0031	Loss: 0.4781 RMSE: 0.6923 Kappa: 0.0000
ResNet34 + LSTM	Hidden Dim: 256 LR: 0.00005 Weight Decay: 0.00003 Smoothness Loss: 0.35386 Consistency Loss: 0.23114 Unfreeze: 75%	Loss: 0.3922 RMSE: 0.7156 Kappa: 0.0200	Loss: 0.5164 RMSE: 0.6922 Kappa: 0.0200
ViT + GRL + Discriminator	LR: 0.0001 Adversarial Weight: 0.05	Loss: 0.0154 F1-Macro: 0.0861	Loss: 0.3141 F1-Macro: 0.0751

Learning Curves

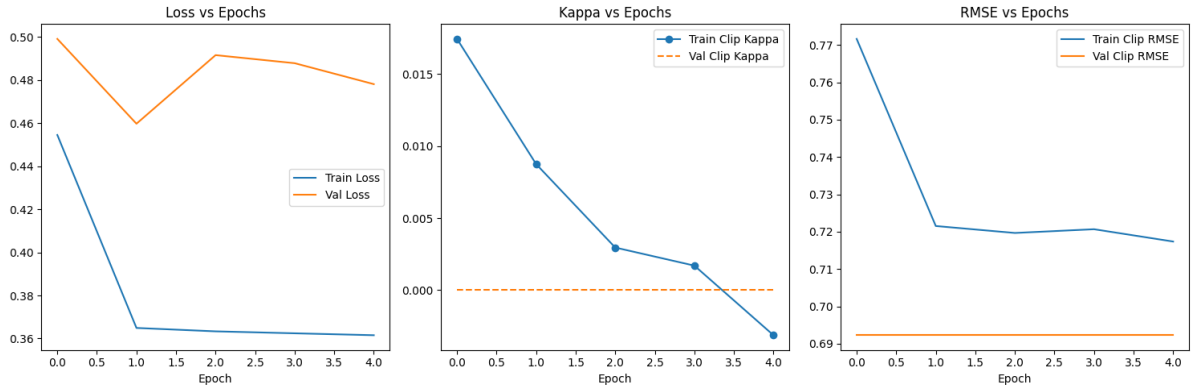

Figure 1: MobileNetV2+LSTM Training and Validation Curves across Epochs (Best Hyperparameter Combination).

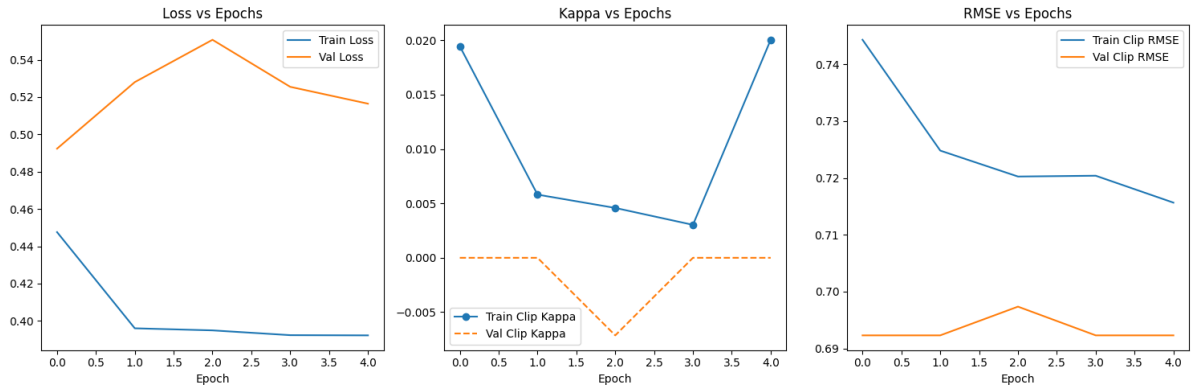

Figure 2: ResNet34+LSTM Training and Validation Curves across Epochs (Best Hyperparameter Combination)

4 Models' testing and evaluation

4.1 Testing

Describe testing process:

- Test set size: Total clips: 1634, Percentage: 20.64%.
- Metrics used: F1-Score, Kappa, Accuracy, Precision and Recall.
- Decision threshold: In one of our attempts to suppress majority classes domination, we made the threshold of a majority class 3x times higher than that of a minority class.

Evaluation metrics

- In Classification attempts: Accuracy, Precision, Recall, F1-score, Confusion Matrix
- In Regression attempts: RMSE, Kappa

Table 4: Values for the evaluation metrics achieved on the test set for the best model from each architecture.

Architecture	**Best combination of hyper-parameter values**	**Values obtained for evaluation metrics**
MobileNetV2 + LSTM	Hidden Dim: 128 LR: 0.00002 Weight Decay: 0.00033 Smoothness Loss: 0.12151 Consistency Loss: 0.00427 Unfreeze: 50%	Accuracy: 0.5184 F1-Macro: 0.1707 F1-Weighted: 0.3540 Kappa: 0.0
ResNet34 + LSTM	Hidden Dim: 256 LR: 0.00005 Weight Decay: 0.00003 Smoothness Loss: 0.35386 Consistency Loss: 0.23114 Unfreeze: 75%	Accuracy: 0.5184 F1-Macro: 0.1707 F1-Weighted: 0.3538 Kappa: 0.0
ViT + GRL + Discriminator	LR: 0.0001 Adversarial Weight: 0.05	F1-Macro: 0.0455 F1-Weighted: 0.5425

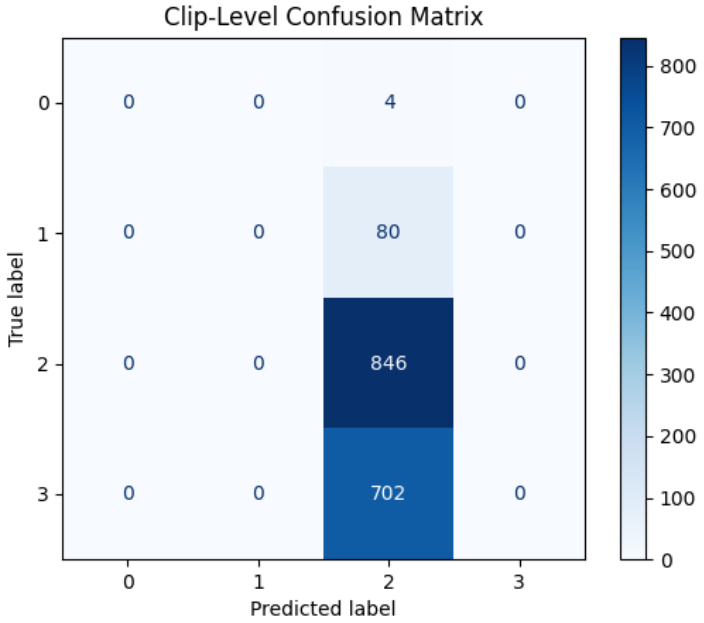

Figure 3: Confusion matrix for both MobileNetV2 and ResNet34.

4.2 Over/under-fitting assessment

Results have shown clear underfitting, which was expected due to the limited data size of 9000 units (clips) over all splits, combined with the sever imbalance, the models in general over all architectural combinations, hyper parameters and settings collapsed early in training and had clear bias toward the majority class 3 and 2 especially the latter one.

4.3 Results analysis

- The Adversarial ViT with its GRL and Discriminator was the closest to mitigate the difficulties introduced by this benchmark dataset. Mainly due to the attention-based architecture that the other CNN-based models lack.
- Setting and tuning different hyperparameters like the learning rate and the regularization terms had clear effect in boosting the models' performance and survival against this challenging data set.
- Failure cases: Most of our model combinations fell short within several epochs and couldn't surpass the early stopping term, since they figure out early that blindly choosing predicting the majority classes will result in the highest overall performance due to the classes' distribution.

4.4 Effectiveness assessment

- Number of parameters:
  - MobileNetV2 + LSTM: 2.94 Million trainable parameters in its best state (50% frozen backbone).
  - ResNet34 + LSTM: 22.08 Million trainable parameters in its best state (25% frozen backbone).
  - Adversarial ViT: 296 Thousands trainable parameters in its best state (fully frozen backbone).
- Model Size on Disk:
  - MobileNetV2 + LSTM: 12.07 MB.
  - ResNet34 + LSTM: 88.5 MB.
  - Adversarial ViT: 344 MB.
- Training Time:
  - MobileNetV2 + LSTM: 6 Hours.

- ResNet34 + LSTM: 8 Hours.
- Adversarial ViT: 9 Hours.

- CPU-based Inference Speed:
  - MobileNetV2 + LSTM: 55 FPS.
  - ResNet34 + LSTM: 20 FPS.
  - Adversarial ViT: 5 FPS.

4.5 Interface development

We developed a simple UI using Gradio, where users can submit a video and get his result.

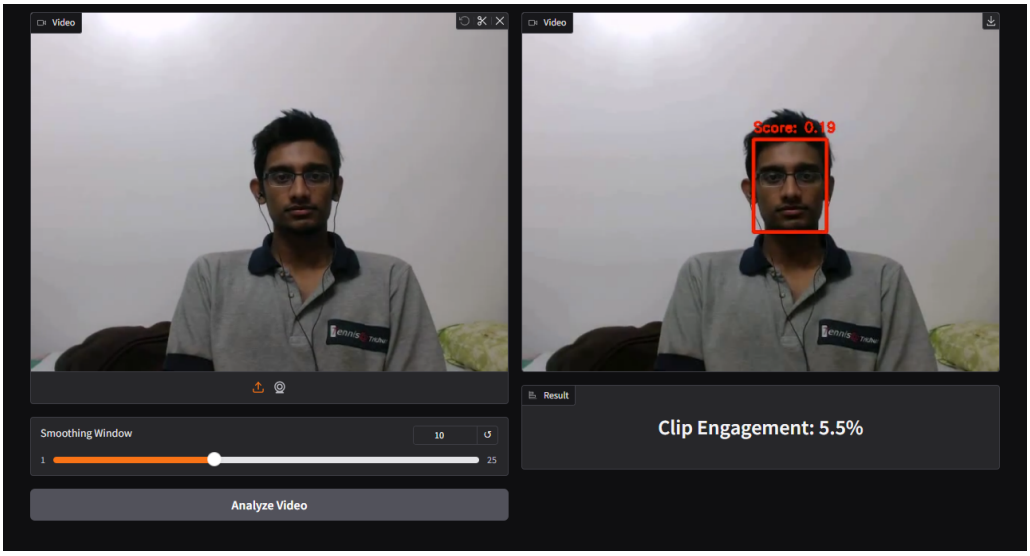

Figure 4: Interface demo: input clip and output prediction.

4.6 Critical evaluation of models

- Data improvements: The most critical factors in limiting performance, with the current set having "in-the-wild" settings over a very small size of data and extreme class imbalance, the results couldn't have gotten any better because the main issue lie within the dataset. So any extra data with more stable background settings and balanced labeling would solve the issue completely.

- Model improvements: The implemented deep learning model will help the instructor understand the student's educational situation, and thus treat each student according to his individual circumstances, thus, introducing a new marking mechanism in the educational system.

An improvement to the system would be using smaller models to reduce the memory and computational time while maintaing high level accuracy, because we suffered a lot of the long execution time, also using techniques to let the system detect more than one face in the video, one last improvement is to use different dataset since it is "In-the-Wild" Setting.

- Deployment: The implemented models is computationally heavy. On a GPU it can process short clips in near real time, but on CPU-only systems real-time inference is not feasible.

  For classroom deployment scenarios where multiple students must be processed simultaneously, this model would not scale efficiently without GPU acceleration.

  Thus, the current solution meets accuracy requirements but does not fully meet real-time constraints for low-resource devices.

References

[1] Abhay Gupta, Arjun D'Cunha, Kamal Awasthi, Vineeth Balasubramanian, AUGUST 2015 *DAiSEE: Towards User Engagement Recognition in the Wild*. Journal, Pages(4-11).

[2] Ghani, M.U., Ali, S. and Humayun, M. (2021) 'Improving state-of-the-art in Detecting Student Engagement with Resnet and TCN Hybrid Network', 2021 IEEE 18th International Conference on Smart Communities: Improving Quality of Life using ICT, IoT and AI (HONET), pp. 119-124.

[3] Architecture Reference:Bertasius, G., Wang, H. and Torresani, L. (2021) 'Is Space-Time Attention All You Need for Video Understanding', Proceedings of the International Conference on Machine Learning (ICML).

[4] DAiSEE Application: Liao, J., Liang, Y. and Sun, J. (2022) 'Deep Video-Based Engagement Recognition in the Wild', IEEE Access, 10, pp. 120-130. (Uses transformer-based temporal modeling).