
# Lie Detection
# Through Facial Micro Expressions

Final Year Project Report

**Project Supervisor**

Dr. Muhammad Atif Tahir

**Project Team**

Junaid Sayani 21K-3401
Bilal Hassan 21K-4669
Daniyal Khan 21K-4831

Department of Computer Science

***National University of Computer & Emerging Sciences***
Karachi Campus

May 2025

| | |
|---|---|
| Project Supervisor | Dr. Muhammad Atif Tahir |
| Project Team | Junaid Sayani 21K-3401<br>Bilal Hassan 21K-4669<br>Daniyal Khan 21K-4831 |
| Submission Date | |

**Dr. Muhammad Atif Tahir** _____
***Head Of School***

**Dr. Zulfiqar Memon** _____
***Director FAST***

FAST School Of Computing

National University of Computer and Emerging Sciences
Karachi Campus
December 2024

# 1 Acknowledgement

To begin with, we would first like to thank the Almighty God for making us capable enough to work on this project and complete it within the given time constraint, with all requirements being met.

We want to extend our heartfelt gratitude to Dr. Muhammad Atif Tahir. This project would not have been possible without his continuous support, patience, motivation, and frequent sharing of their field expertise to solve all the problems we faced during the project. Not just academic support but Dr. Atif Tahir supported us with all of our technical needs, too, by giving us access to work on gpu's, and high-end pc's in his Video Surveillance Lab. I would also like to acknowledge the financial support provided by our university, FAST-NUCES, for providing us with the required resources for our projects.

I am also grateful to my fellow lab assistants, instructors, and the honorable Jury for taking their time to review our project and keeping us resourceful throughout it. The assistance from all the individuals mentioned above and the constant support from our family members has made this journey meaningful.

## 2 Abstract

In high-stakes environments such as police interrogations, the accurate detection of deception is crucial for maintaining the integrity of investigative procedures. Traditional methods, relying on verbal cues and intuition, often lack objectivity and reliability. This project introduces a video-based deception detection system that leverages advanced deep learning techniques to analyze micro-expressions and facial action units (AUs) as indicators of deception. Utilizing models such as Constrained Local Neural Field (CLNF), Convolutional Neural Networks (CNNs), and Long Short-Term Memory Networks (LSTMs), the system achieves robust performance in analyzing facial expressions captured through low-resolution video feeds.

Building upon this foundation in FYP-II, we integrated ensemble learning techniques, most notably bagged LSTM models, which significantly outperformed baseline models acheiving an accuracy of 92% and enhanced generalizability. We also uncovered a critical flaw in the base paper's data-splitting strategy, which we addressed to reduce overfitting and improve evaluation fairness. In parallel, we proposed a novel approach involving the K600 Spatio-Temporal Transformer, trained directly on raw video frames. Although this transformer-based pipeline achieved slightly lower accuracy than our LSTM ensembles (80%), it introduced a unique architecture to this domain and eliminated the intermediate AU extraction step, thus simplifying deployment. Bagging was also applied to transformer models, further improving stability. Designed for scalability and ethical compliance, the final system accommodates legal and cultural considerations, ensuring fairness and privacy in its deployment. This tool has the potential to enhance law enforcement practices by providing an objective means to assess the credibility of suspects during interrogations, contributing to advancements in security, psychology, and AI-driven behavioral analysis.

# Contents

# List of Figures

# 3 Introduction

## 3.1 Background

In high-stakes environments such as police interrogations, detecting deception can be critical for uncovering the truth. Traditional methods of identifying lies rely on verbal cues and intuition, but these can often be misleading. This project addresses the need for an objective, automated solution to deception detection by leveraging advanced computer vision techniques.
The system uses facial expressions, captured through low-resolution cameras, as indicators of truthfulness. By applying deep learning models like CLNF, CNN, and LSTM, the system analyzes micro-expressions and facial action units to identify signs of deception during police interrogations. The solution is designed to be scalable, accurate, and practical for real-world applications, offering law enforcement a reliable tool to support their investigations and enhance the integrity of their procedures.

## 3.2 Scope

The scope of this project is to develop a video-based deception detection system for use in police interrogations. The system will analyze facial expressions and micro-expressions using deep learning techniques, including CLNF + OpenFace, CNN, and LSTM, to detect signs of deception in low-resolution video footage. The focus will be on creating a scalable and reliable tool that can be used in real-world interrogation settings, providing law enforcement with an objective means of assessing the truthfulness of suspects. The scope of FYP-II is centered on advancing the mono-modal deception detection system developed in FYP-I by integrating ensemble learning and transformer-based deep learning techniques. The project focuses exclusively on visual data captured from low-resolution video feeds, particularly facial expressions and micro-expressions during interrogations. It investigates the viability of ensemble methods like bagging to enhance LSTM-based models and explores cutting-edge transformer architectures such as the K600 Spatio-Temporal Transformer as a novel approach. The work intentionally limits itself to non-verbal cues without incorporating multimodal inputs like audio or physiological signals, allowing for scalability and applicability in real-world law enforcement scenarios with minimal hardware requirements.

## 3.3 Aims

- **Enhance model performance using ensemble learning**:

  Implement and evaluate ensemble techniques, particularly bagging, on LSTM models to improve prediction accuracy and model robustness for deception detection.

- **Identify and resolve overfitting issues in previous approaches**:

Analyze the shortcomings in the base paper's data splitting strategy and redesign the evaluation pipeline to ensure fair, non-leaky data division and improved generalizability.

- **Develop and experiment with transformer-based architectures**:

  Fine-tune a K600 Spatio-Temporal Transformer model directly on raw video frames to assess its feasibility as an alternative to traditional LSTM-based methods.

- **Streamline the detection pipeline by removing intermediate steps**:

  Propose a novel approach that bypasses the extraction of facial Action Units, simplifying the preprocessing phase while maintaining competitive accuracy.

- **Evaluate the effectiveness of ensemble learning on transformers**:

  Apply bagging techniques to transformer models and compare their performance against both baseline and bagged LSTM models.

- **Maintain scalability and real-world applicability**:

  Ensure all models and approaches are optimized for low-resolution video inputs, keeping computational requirements realistic for deployment in resource-constrained law enforcement environments.

## 3.4 Intended Audience

The intended audience for this project includes, Tech Firms, Educational Institutions, law enforcement agencies and criminal justice professionals who can benefit from an objective and automated tool for detecting deception during police interrogations. This system is also valuable to researchers and academics in the fields of computer vision, artificial intelligence, and psychology, who are exploring applications of AI in human behavior analysis.

# 4 Related Work

Deception detection is a common challenge in everyday conversations [6], online videos [2,3], and high-stakes situations like court trials [4] and interviews [5]. Many studies have managed to outperform the average person's ability to spot deception [4]. However, much of the research relies on private datasets and focuses on only one type of data [5]. Public datasets like Real-Life Court Trials [4], DDPM [5], Silesian Deception [8], Mu3D [7], and Box of Lies [6] have been released to help train models for deception detection.

Detecting deception in videos is particularly difficult because of the complexity of video data and its mixed nature, the lack of clear cues for deceit, and the limited availability of labeled data [2]. Recent studies show that both verbal and non-verbal behaviors play a role in detecting deception. Verbal cues include features like word patterns and sentence structures [4], while non-verbal cues involve facial expressions and physiological signals such as heart rate and blood oxygen levels [5].

To improve detection, researchers first extract features from the data before using machine learning and deep learning models. Classifiers like Random Forest [6], Support Vector Machines (SVMs) [4], Neural Networks (NNs) [4], Convolutional Neural Networks (CNNs) [2], and Long Short-Term Memory Networks (LSTMs) [3,2] are commonly used. However, raw data is usually preprocessed before being fed into these models. For example, [6] divided videos into segments based on conversation turns and labeled facial movements like eyebrow shifts and gaze. Similarly, [3] broke videos into 30-frame chunks after cleaning the data, leading to better accuracy.

Cross-dataset testing shows that models trained on multiple datasets often perform poorly because of differences in the data [3]. This highlights the importance of fine-tuning parameters like regularization values [2]. Accuracy rates in various studies range widely, with some achieving up to 89.49% on the Real-Life Trials dataset [3] and others as low as 54.5% when using physiological data [5].

Overall, the research shows that combining multiple types of data—such as facial expressions, speech, and physiological signals—can improve deception detection. Methods like FACS and LSTMs have been especially effective for video-based detection. However, differences between datasets remain a major challenge in building systems that work well across different scenarios. Multimodal approaches, like those tested on the Bag-of-Lies dataset, show potential by capturing a mix of verbal and non-verbal signals during deceptive interactions [3,5,6].

# 5 Requirements

## 5.1 Functional Requirements

### 5.1.1 Use Case Diagram

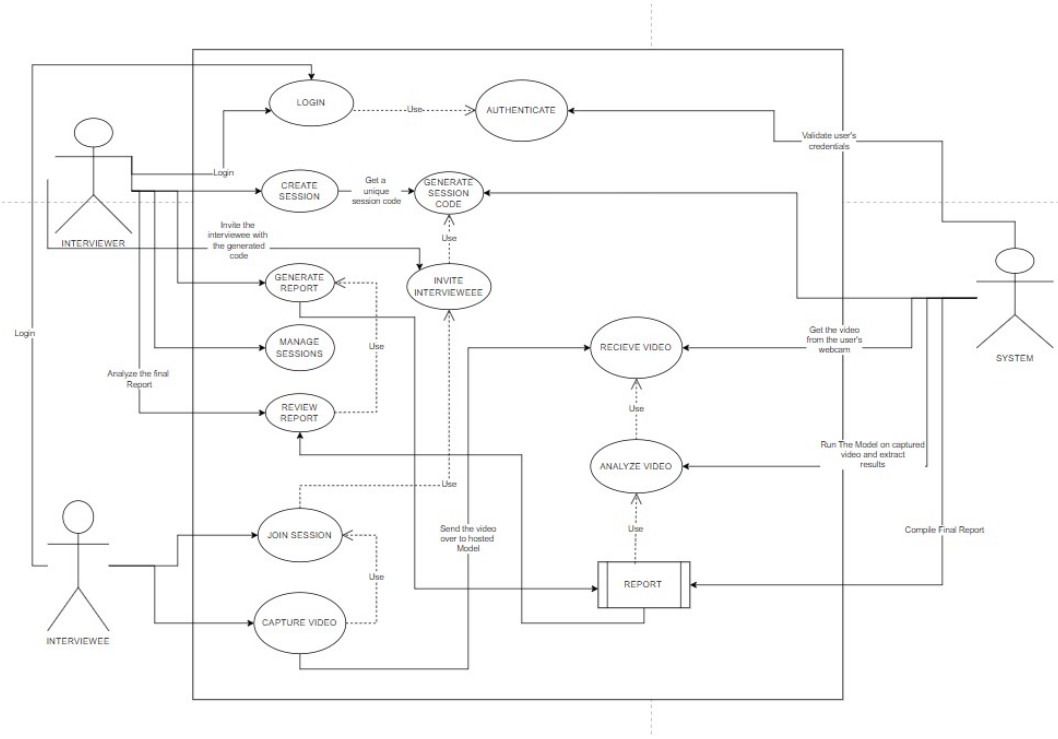



Figure 1: Use Case Diagram

### 5.1.2 Use Case Descriptions

| <Use Case Name> | Interview Video Analysis System | |
|---|---|---|
| **Use Case Id** | UC-01 | |
| **Actors:** | Interviewer, Interviewee, System | |
| **Feature:** | Video analysis system for assessing deception in interview scenarios. | |
| **Pre-Conditions:** | 1. The interviewer and interviewee must be logged into the system.<br>2. Interviewer must have generated a session code and invited the interviewee.<br>3. Video capture equipment must be available and functional. | |
| **Scenarios** | <Happy Path> | |
| **Step#** | **Action** | **Software Reaction** |
| 1. **Join Session** | The interviewee joins the session using the session code. | The system verifies the session code and allows the interviewee to proceed. |
| 2. **Capture Video** | The interviewee captures the video for analysis. | The system records the video and uploads it for processing. |
| 3. **Receive Video** | The system receives the video and prepares it for analysis. | |
| 4.**Analyze Video** | The system runs the facial expression analysis model. | The system extracts results based on the analysis of micro-expressions.(AU's) |
| 5. **Generate Report** | The system compiles a final report based on the video analysis. | The report is prompted to the interviewer |
| 6. **Review Report** | The interviewer reviews the analysis report. | The system allows the interviewer to provide feedback or close the session. |
| **Alternative Scenarios** | **Error Conditions** | |
| **Step#** | **Description** | |
| *a. | System Failure at any point | *a.1. Close the session *a.2. Generate Intermediate Report. *a.3. Generate coresponding notifications. |
| 1a. | Invalid Session Code Entered | 1a1: Prompt user with error. |
| 2a. | No camera detected | 2a.1: Prompt user to allow permission for camera recording. |
| 2b. | Video capture failure | 2b.1: Prompt user to remove any objects from infront of the camera. |
| 4a. | Model failed to return AU's | 4a.1: Try from the next batch<br>4a.2: Prompt the user to show face clearly |
| **Use Case Cross Referenced** | UC-02 | |

Figure 2: UseCase Desc-1

## 5.2 Non-Functional Requirements

## 5.3 Performance Requirements

- The system will maintain an overall detection accuracy of at least 70%, given unobstructed facial input.
- The system will be scalable to handle multiple concurrent interview sessions without a significant drop in performance.
- The system will be highly reliable and operational during active interview sessions, with an uptime of at least 99%.

## 5.4 Usability Requirements

- The user interface will be simple, intuitive, and user-friendly for both interviewers and interviewees, requiring minimal technical knowledge to operate.
- Feedback (e.g., results, warnings, or errors) will be displayed clearly and in real-time for interviewers.
- The system will ensure fairness, avoiding bias or discrimination in deception detection results, particularly concerning different demographics or cultural groups.

## 5.5 Security Requirements

- User authentication will be enforced, ensuring that only authorized users (interviewers and interviewees) can access the system and initiate sessions.
- All session data (including video recordings and results) will be securely stored and encrypted to protect privacy and sensitive information.
- The system will comply with data protection regulations, ensuring that personal and video data is handled responsibly and stored securely.

## 5.6 User Documentation

- A user manual will be delivered to ensure the admin, as well as the meter readers, are well aware of how the application works. This user manual will also be helpful in case someone faces an issue in running the application.
- Reliable Easy-to-use UI to guide through the system (Context Sensitive Help)

# 6 System Design

## 6.1 Architecture

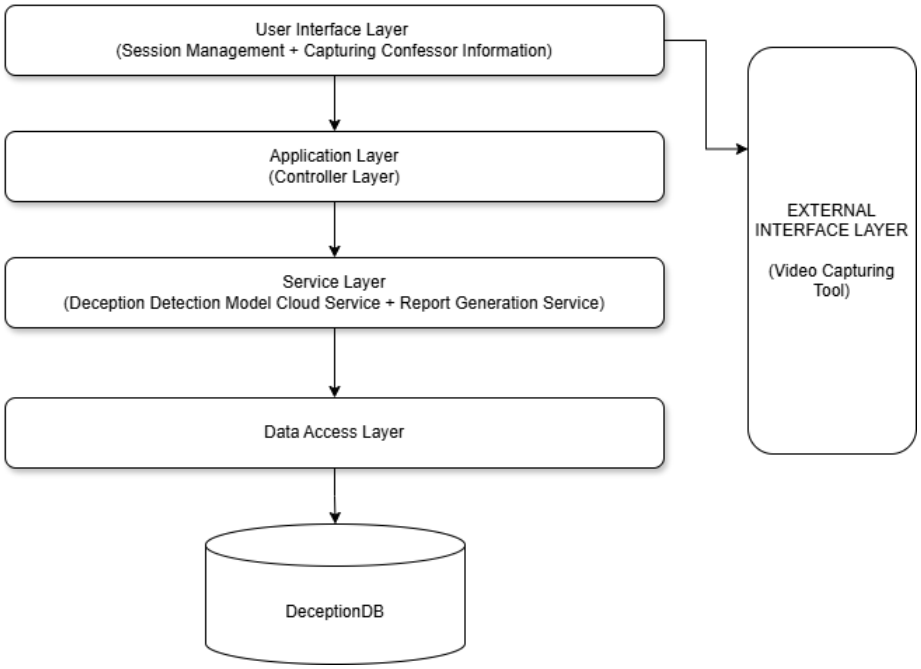



Figure 3: Architecture

## 6.2 ER Diagram

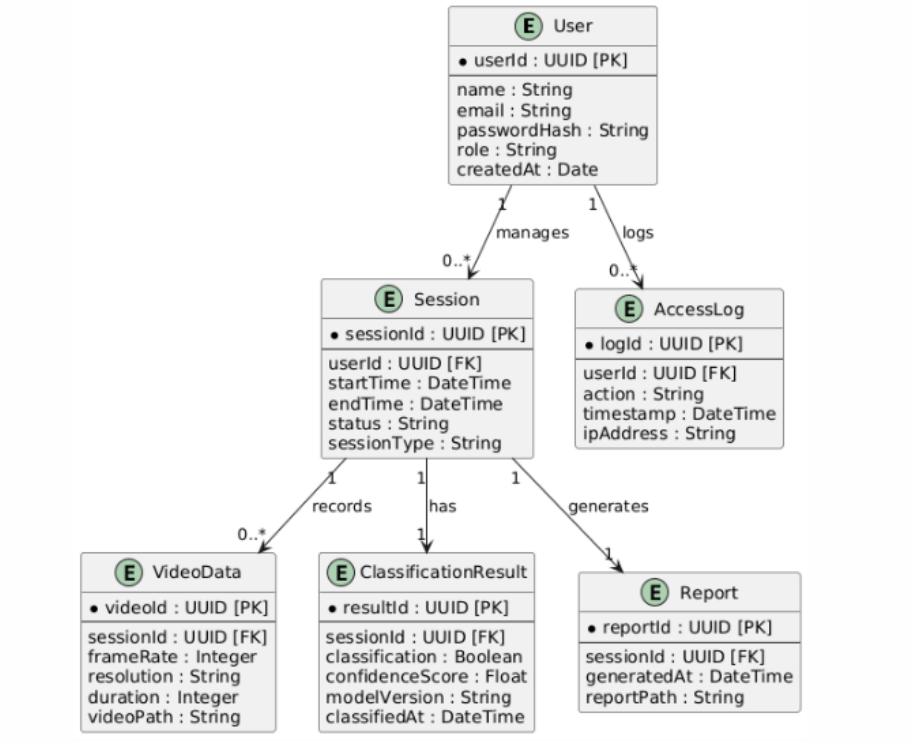



Figure 4: ER Diagram

## 6.3 Sequence Diagram

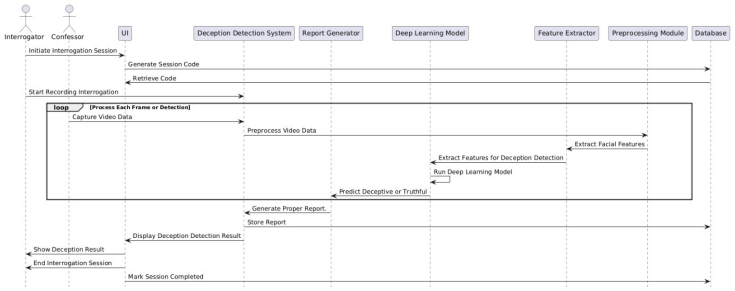



Figure 5: Sequence Diagram

## 6.4 State Diagram

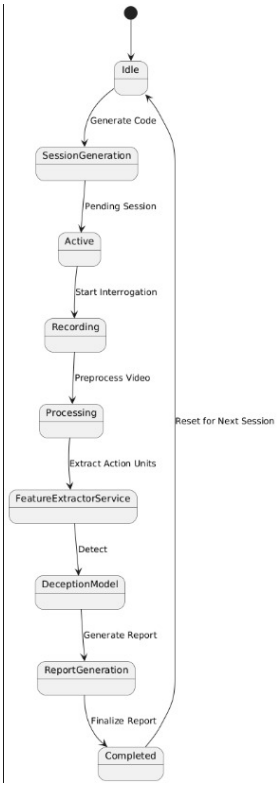



Figure 6: State Diagram

# 7 Methodology

## 7.1 Datasets Used

**Real-Life Court Trials (RLCT):** The Data shown here in **Figure-7**, is a real-time captured data in a high-stake environment setting, that is a court room. There are some guilty and some innocent confessors in the court room, and the labels of truth and deceiving is assigned on that basis only.

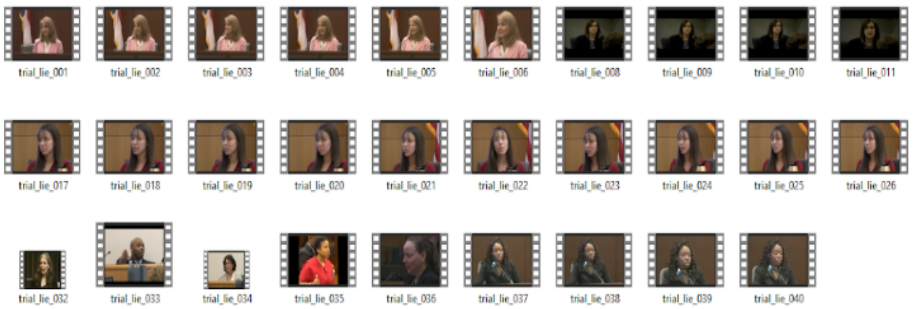



Figure 7: RLCT DS

**Silesian:** The Dataset in Figure-8 is a dataset recorded in a controlled lab environment. Where the subject is shown multiple images, and they have to follow an order of sequences, in providing their opinions about that image (i.e Agree by confirmation, Deceive by confirmation, Agree by negation etc)

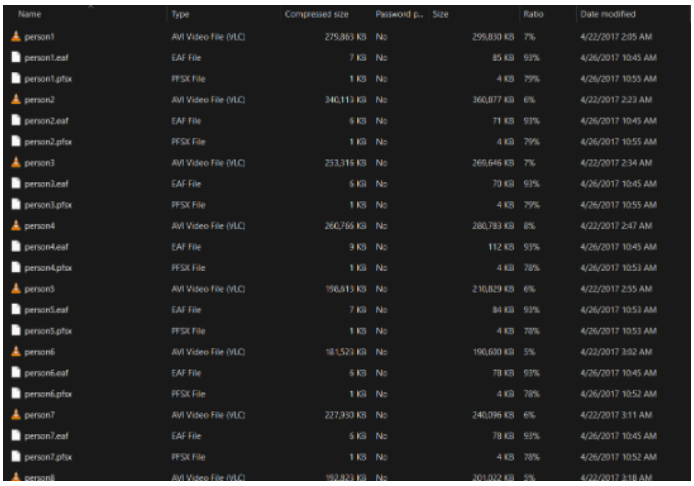



Figure 8: Silesian DS

## 7.2 Pre-Processing and Data Cleaning

**Removed Low-Quality Data**: Excluded videos with excessive noise, low resolution, or poor lighting conditions to ensure consistent input quality.

**Dealt with Missing Frames**: Identified and removed incomplete or corrupted frames that could disrupt the training process.

**Contrast Enhancement**: Enhanced the visibility of facial features by adjusting the contrast of each frame.

**Filtered Irrelevant Segments**: Removed video segments without relevant facial data (e.g., obscured faces or unrelated actions).

**Balanced Dataset**: Ensured a balanced representation of both deceptive and truthful instances to avoid bias during training.

**Video Deletion:** Deleted Videos where no considerable output was achieved in facial expressions

.

**Deception By Confirmation:** Segmented, separated and removed all the deception by confirmation confessions in Silesian Dataset, since it had some issues according to the author.

## 7.3 Data Segmentation & Extraction

### 7.3.1 Annotations

The annotations for silesian datasets are given as combination of action-units which depicts some sort of emotion, and also the time stamps for each confession within the video. Since we ourselves are calculating the Action-Units all we need is the timestamp for each confession so that we can extract them and sort them accordingly. The file for annotations + video is a MATLAB file, and upon opening it looks something like this (shown in **Figure-9** below)

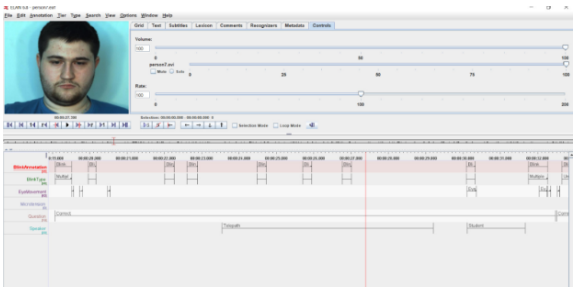


Figure 9: Annotation Silesian

## 7.4 Data Chunking and Architecture

Firstly, all the confessions were sent to OpenFace library which through a Pre-trained Constrained-Local-Neural-Field generated 35 Action Units and also the several various landmarks. We then narrowed down to the final 32 Action Units for each frame, that would aid us in detecting lies.

The next step involved in this pipeline was to divide these confession videos into training and testing folders. We broke down our data in HOOCV (Hold out cross validation) in this ratio: 70:15:15, for Training, Validation and Testing respectively.

Now, we break our videos into chunks of 30 frames each. And we assigned each chunk the output label corresponding to the original video label. (i.e If the video was truthful all of its chunks, are assigned the label Truth). This gives a final feature vector of 30x32 for each separate chunk with its output label, which is now ready to be processed in our LSTM-MLP setup.

These chunks proceed into a sequential LSTM layer which captures the temporal dependencies among the frames of each chunk. Consider **xt** as the input frame that is (1x32) feature vector at time = t, the input first updates the forget gate, input gate and candidate gate as follows:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right) \quad (Forget - gate) \tag{1}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \quad (Input - gate) \tag{2}$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right) \quad (Candidate - cell - state) \tag{3}$$

After these updates, there is an internal cell state update (memory update) using the previously calculated candidate state, this is important for capturing long term temporal dependencies:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad (Cell - state - update) \tag{4}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \quad (Output - gate) \tag{5}$$

And finally, the hidden-state is updated using the output gate (that carries the sequential motion, and the memory state:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (Hidden - state - update) \tag{6}$$

The final state at time t=n is given as *ht* which we proceed into a Dropout layer (where D=0.17), this is to regularize the model and prevent over-fitting. Which then heads into a Fully Connected Dense layer to predict the chunk as

Truth or Deceptive. To backpropogate and update the weights we use the Binary Cross Entropy Loss Function (1), since it is a classification problem.

$$L = -\frac{1}{N}\sum_{i=1}^{N} \left[ y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right] \tag{7}$$

Where:

- **N** is the total number of samples.
- **yi** is the ground-truth label of sample.
- **yi capˆ** is the predicted label.

For Silesian Dataset we again followed the same pipeline, as defined in the base paper we were following. However, there was one additional step included that was of automating the confession extraction, unlike Court Trials, here each video had 10 different confessions with its annotations, so we had to write code on MATLAB that would automate the extraction each of these confessions from the videos.

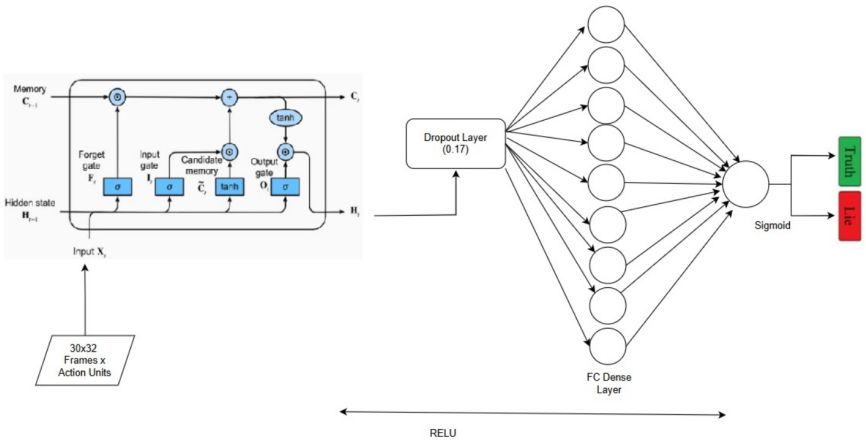



Figure 10: Model

## 7.5 Ensemble Learning Approaches

**Bagging (Bootstrap Aggregating):** We train multiple LSTM models on different subsets of the dataset, each learning independent patterns in deception cues. The final prediction is determined via majority voting:

$$\hat{y} = mode\{f_1(X), f_2(X), ..., f_n(X)\} \tag{8}$$

where $f_i(X)$ represents the prediction from the $i$-th LSTM model.

**Boosting:** Boosting is employed by training LSTMs sequentially, where each model focuses on correcting the errors made by its predecessor. The process involves:

1. Training multiple LSTM classifiers sequentially.
2. Assigning higher weights to misclassified samples after each iteration.
3. Using a weighted sum of predictions from all models to make the final classification.

The final classification score is computed as a weighted sum of individual model outputs:

$$\hat{y} = \sum_{i=1}^{N} \alpha_i f_i(X) \tag{9}$$

where $\alpha_i$ represents the weight assigned to model $i$ based on its accuracy.

**Stacking:** Stacking combines the predictions of multiple LSTMs by training a meta-classifier on their outputs. Unlike Bagging and Boosting, where models work independently or sequentially, Stacking learns how to optimally combine multiple models' outputs. The process involves:

- Training multiple LSTM classifiers $(f_1, f_2, ..., f_N)$ on the dataset.
- Collecting their predictions as input features for a secondary model.
- Using a **meta-classifier** (an LSTM in our case) to learn the best combination of predictions.

The final classification is given by:

$$\hat{y} = g(f_1(X), f_2(X), ..., f_N(X)) \tag{10}$$

where $g$ represents the meta-classifier that learns from the base models' outputs.

## 7.6 Fine-Tuning Vision Transformer

### 7.6.1 TimeSformer (ViT-Based Model)

In contrast to LSTM-based models, we employ a TimeSformer architecture (as shown in Figure.11) to leverage the power of self-attention in both spatial and temporal dimensions. The model processes video frames as sequences of non-overlapping patches, capturing both short-term spatial dependencies and long-range temporal relationships. The pretrained TimeSformer is fine-tuned on our dataset with modifications to its temporal embeddings for shorter video sequences.

**Pretraining and Fine-Tuning** We use a pretrained TimeSformer from Kinetics-600 and adapt it for deception detection:

- Adjusting temporal embedding layers to accommodate **8-frame** video sequences. (Pre-Trained ViT had 96-Frame sequence)
- Replacing the classification head for binary classification (deceptive vs. truthful).
- Fine-tuning the model to optimize deception-related feature extraction.

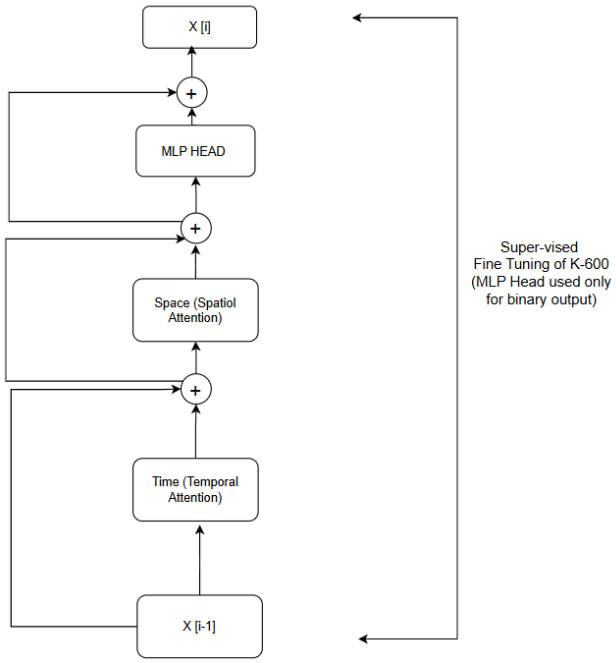



Figure 11: ViT Architecture

## 7.7 Transformers as an alternative to LSTM

To explore transformer architectures in a more controlled and foundational manner, we also implemented a simple transformer model as a direct replacement for LSTMs in our deception detection pipeline. Unlike recurrent models, which process data sequentially, the transformer operates on the entire sequence in parallel, enabling faster training and superior modeling of long-range dependencies. In our approach, facial features extracted from each frame using CLNF were first flattened into a sequential format and then fed into the transformer encoder.

The key component of this model is the self-attention mechanism, which allows the network to weigh the relevance of different time steps within the sequence. The self-attention is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

Here, $Q$, $K$, and $V$ represent the query, key, and value matrices, and $d_k$ is the dimension of the key vectors. This formulation enables the model to attend to critical facial expression patterns that may be temporally distant but contextually significant for deception.

The output from the transformer encoder is passed through a classification head to determine whether the behavior is truthful or deceptive. Although this simple transformer model did not outperform our LSTM ensemble, it achieved promising results with competitive accuracy and offered a cleaner, end-to-end architecture without the complexities of sequence recurrence. The success of this model reinforces the growing potential of transformer-based architectures in behavioral and affective computing tasks.

## 7.8 Extracting Meaning from Action Units

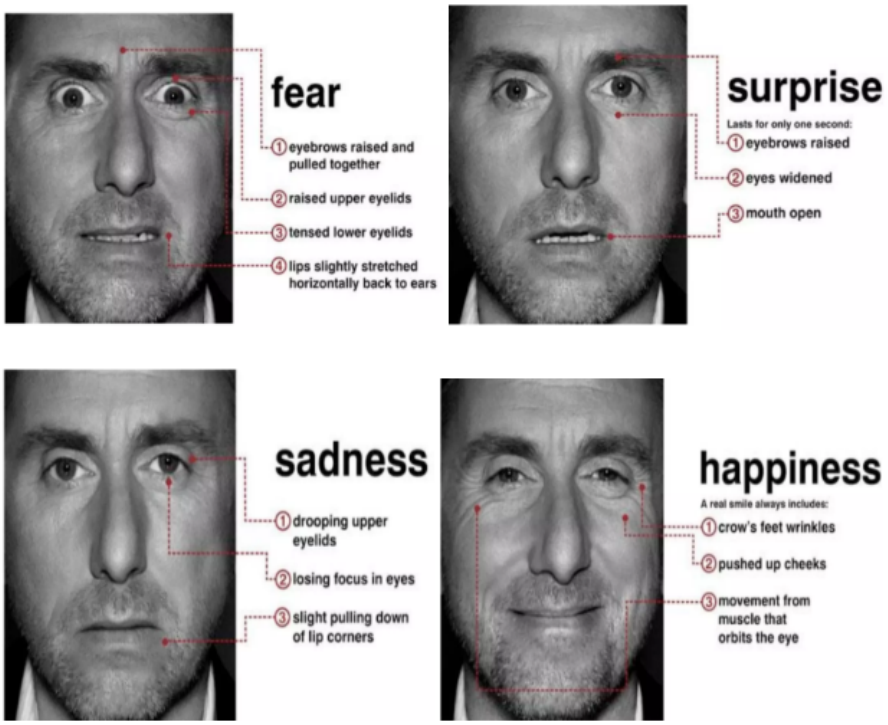



Figure 12: AU

**Action Units (AUs)** as shown in **Figure-12** play an important role in conveying emotions and contributing to the overall mood and feeling expressed through facial expressions. Each AU represents the activation of a specific facial muscle or group of muscles, such as raising the eyebrows, tightening the lips, or widening the eyes. These micro-movements, when combined, form the basis of complex emotional expressions like happiness, sadness, anger, or surprise. For example, the activation of AU12 (lip corner puller) is often associated with a smile, which contributes to a positive mood, while AU4 (brow lowerer) is linked to expressions of anger or determination. By analyzing these subtle changes, it becomes possible to infer not only the immediate emotional state of an individual but also their underlying mood, providing valuable insights into their feelings and intentions.

## 7.9 Model Architecture's Code Snippets

The Model's Architecture is shown in the code snippet below, in Figure-13 . Which represents a single LSTM Layer, followed by a dropout layer with (D=0.17). And finally a fully connected Dense Layer with Sigmoid Activation (since output is in binary).

The Learning Rate we used was LR=0.001 and the loss function was "Binary-Cross Entropy with ADam Optimizer. We kept the batch size fixed at 50, with epochs at 500.

```
model = Sequential()
model.add(LSTM(170, input_shape=input_shape))
model.add(Dropout(0.17))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer=Adam(learning_rate=0.001),metrics=['accuracy'])

history = model.fit(X_train,y_train,batch_size=50,epochs=500, verbose=1, validation_data=(X_val,y_val))
```

Figure 13: Model Architecture

Figure 14: OpenFace AU Extractionn

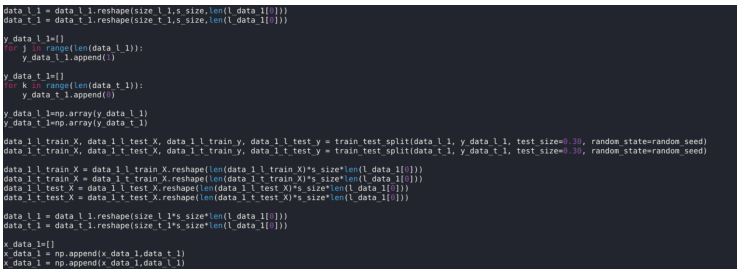


Figure 15: Train-Test Split

# 8 Testing and Results

## 8.1 Results on Base Paper's Architecture (Simple LSTM)

| Training Dataset | Testing Dataset | Accuracy |
|---|---|---|
| RLCT | RLCT | 86% |
| Silesian | Silesian | 88.7% |
| RLCT | Silesian | 50.9% |
| Silesian | RLCT | 57.58% |

Table 1: Accuracy and - Base Paper

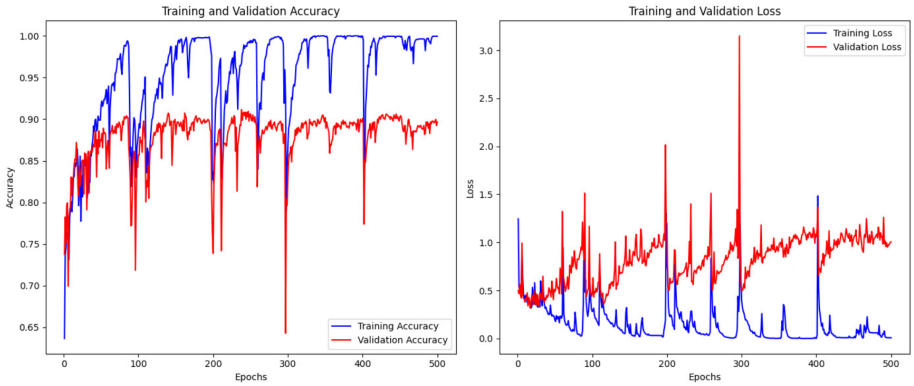



Figure 16: Court Trials Graph

## 8.2 Results on our approaches

### 8.2.1 Bagging - Boosting - Stacking - Transformers - ViT

| | Bagging | Boosting | Stacking | Transformer | ViT |
|---|---|---|---|---|---|
| RLCT | 92.11% | 63.32% | 90.83% | 88.71% | 80.17% |
| Silesian | 89.9% | 60.86% | 85.42% | 59.69% | 81.2% |

Table 2: Table of accuracies for best ensemble techniques

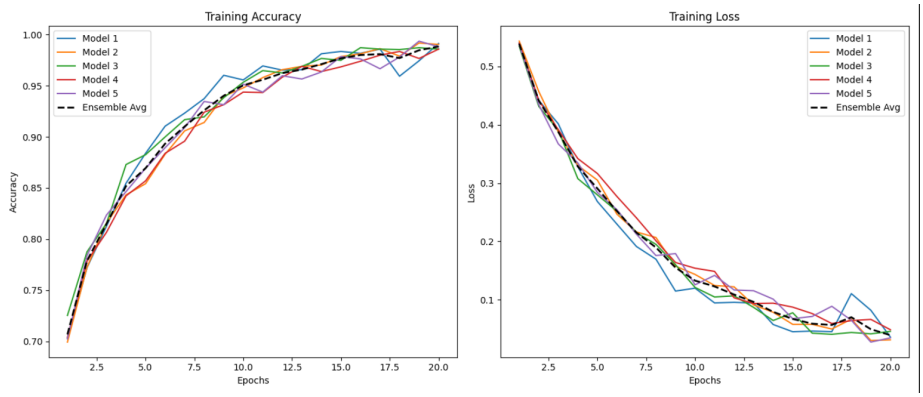



Figure 17: Real Life Court Trials - 5 Model Bagging - Best Performer Example

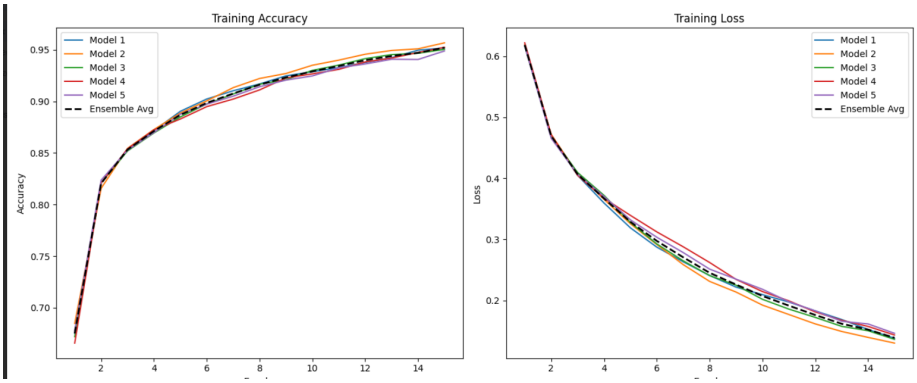



Figure 18: Silesian - 5 Model Bagging- Best Performer Example

# 9 System Diagram

The **System Diagram** shown in **Figure- 19** illustrates the workflow of a deception detection application, integrating various technologies and processes. The system begins with an interrogator accessing the user interface, developed using HTML, CSS, and JavaScript, to either search session history or initiate a new session. For a new session, the interrogator performs a sign-in and verification process, which generates a unique session code. This session code is then processed via a Google API Gateway, linking the front-end with back-end services hosted on Google Cloud. Upon entering the session code, the system retrieves the corresponding session state. The next step involves loading the session data and initializing the calibration process. This includes face alignment, video capture, and pre-processing of the video input. The system utilizes a trained deep learning model to process these inputs. The video data undergoes several stages of analysis, including facial landmark detection, gaze estimation, head pose detection, and facial action unit recognition. These features are fused and normalized for dimensionality reduction to enhance the model's prediction accuracy. The processed data is classified to determine whether the individual is being truthful or deceptive. The outcome is stored in a PostgreSQL database, with reports generated and saved against the session code. At the session's end, a detailed report is generated, displaying whether the responses were classified as truth or lie. The interrogator can retrieve and review these reports for analysis, providing a seamless integration between video analysis and user interaction.

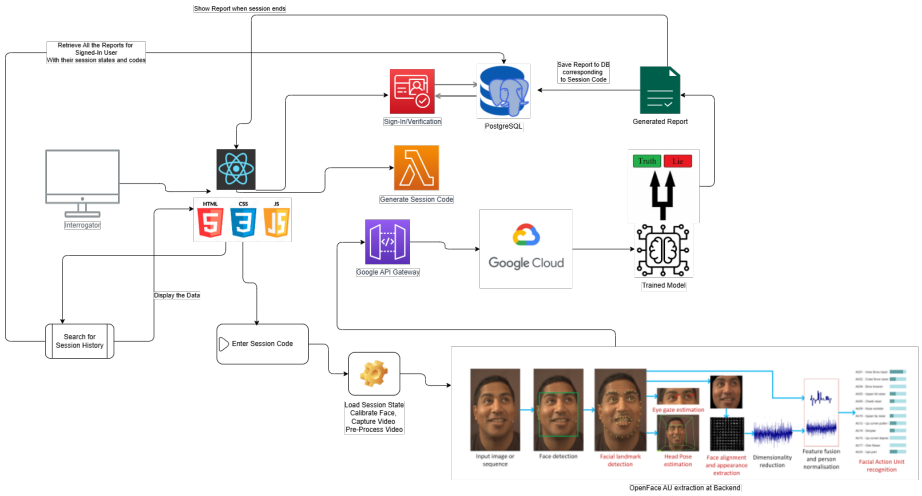
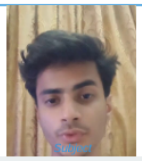



Figure 19: System Diagram

# 10 Example of a Final Report

## Deception Analysis Report



**Report Information**

Generated on: 2025-05-01 03:45:48

### Video Information

**File Details**

File: 20250501_034420_ca41103d.mp4

### Deception Analysis Results

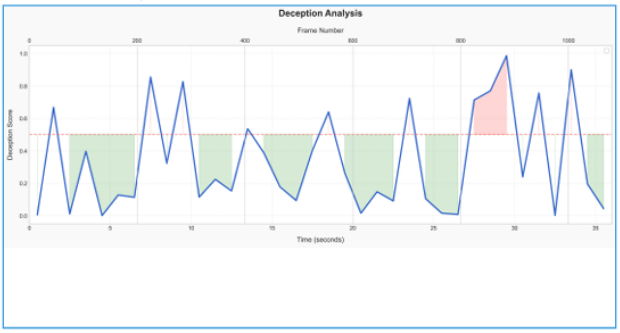



*Figure 1: Deception Analysis Timeline*

### Analysis Summary

| Metric | Value |
|---|---|
| Truthful Periods | 69.4% |
| Deceptive Periods | 30.6% |
| Average Deception Score | 0.33 |
| Average Confidence | 0.65 |

# 11 Conclusion and Findings

In conclusion, the project demonstrated the potential of using facial Action Units (AUs) to detect deception through deep learning techniques. By applying a combination of OpenFace and CLNF for AU extraction, followed by LSTM-based modeling, we were able to achieve promising results in both training and testing. To evaluate the effectiveness of different deep learning architectures in deception detection, we conducted extensive experiments across two datasets: the Silesian Deception De- tection dataset and the Real-Life Court Trials (RLCT) dataset. Our experiments involved multiple LSTM-based ensemble techniques (Bagging, Boosting, and Stacking), a standard Transformer model, and a Vision Transformer (ViT) based TimeSformer architecture.

The performance of LSTM ensembles comprising three and five models, respectively. Among the ensemble techniques, Stacking and Bagging consistently outperformed Boosting across both datasets. Notably, Bag- ging with five LSTM models achieved the highest accuracy overall,92.11% on RLCT and 89.9% on the Silesian dataset. This is to the best of our knowledge, the highest accuracy any other paper has achieved with a mono-modal approach. This also highlights the effectiveness of model variance introduced through bootstrap sampling in capturing subtle deception cues. Vision transformers demonstrated promising results, achieving 81.2% accuracy on the Silesian dataset and 80. 17% accuracy on RLCT. This marks a novel approach to this problem which hasn't been explored by any of the previous paper, which includes fine tuning a spatio-temporal ViT. These results suggest that ViTs can generalize well to deception-related facial patterns, although their performance did not surpass the best LSTM ensemble configurations. The standard Transformer model achieved 59.69% on the Silesian dataset and 88.71% on RLCT,showing better performance on real-world, noisy data but relatively lower accuracy on controlled data.

In conclusion, while Transformer-based models offer scal- ability and modeling power for temporal sequences, the Bag- ging LSTM ensemble remains the most effective strategy in our setup for deception detection using facial action units, particularly in both high-stakes and controlled environments.

# 12 Recommendations For Future Work

In the future, the deception detection system can be enhanced by expanding its capabilities to include multi-modal data analysis, such as voice tone and body language, alongside facial expressions, to improve accuracy in detecting deception. Further research can focus on optimizing the system for real-time processing in diverse interrogation environments, including low-light and noisy settings. Additionally, the model can be trained on a larger and more diverse dataset to improve its generalization across different ethnicities, genders, and age groups.

# 13 Bibliography

# References


[1] V. S. Ramachandran, "Microexpression and Macroexpression," in *Encyclopedia of Human Behavior*, Academic Press, 2012.

[2] H. Karimi, J. Tang, and Y. Li, "Toward end-to-end deception detection in videos," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, 2018.

[3] H. U. Din Ahmed, U. Ijaz Bajwa, F. Zhang, and M. W. Anwar, "Deception Detection in Videos using the Facial Action Coding System," *May 2021.*

[4] V. Pérez-Rosas, M. Abouelenien, R. Mihalcea, and M. Burzo, "Deception Detection using Real-life Trial Data," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (ICMI '15)*, Seattle, WA, USA, 2015.

[5] N. Vance, A. K. Hughes, C. K. P. Shum, and M. J. M. Smith, "Deception Detection and Remote Physiological Monitoring: A Dataset and Baseline Experimental Results," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 4, no. 4, Oct. 2022.

[6] F. Soldner, V. Pérez-Rosas, and R. Mihalcea, "Box of Lies: Multimodal Deception Detection in Dialogues," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, MN, USA, Jun. 2019.

[7] E. Paige Lloyd, Jason C. Deska, Kurt Hugenberg, Allen R. McConnell, Brandon Humphrey, and Jonathan W. Kunstman, "Miami University Deception Detection Database," in *Graduate Student Scholarship*, Miami, 2017.

[8] K. Radlak, M. Bozek, and B. Smolka, "Silesian Deception Database: Presentation and Analysis," in *Proceedings of the 2015 ACM Workshop on Multimodal Deception Detection*, 13 November 2015.