# Multi-Modal Deception Detection using Real-Life Trials Videos

## I. Problem and Motivation

Our project addresses the challenge of enhancing deception detection accuracy by leveraging multi-modal analysis on real-life trial videos. We aim to differentiate truthfulness in courtroom settings by evaluating both verbal and non-verbal cues exhibited by individuals during their testimonies. The significance of this research stems from the need for more accurate and reliable methods to detect deception in legal contexts, where current methods often rely heavily on personal judgements and may fail to consistently identify deceptive behavior.

## II. The Dataset and Method

This dataset contains high-quality video recordings of court trials, sourced from public materials. It covers different trial outcomes like guilty, not guilty, and exoneration. Deceptive videos are those where suspects deny crimes they're later found guilty of, while truthful videos include suspects answering questions about facts confirmed by the police. Witness statements verified by police investigations are labeled as truthful, whereas statements supporting a suspect found guilty are marked as deceptive. Statements from exoneration cases are considered truthful. The dataset has 61 deceptive and 60 truthful videos. Some data examples belong to the same person and same trial. 51 people have only one video, the distribution of others as follows: 1 person 18 videos, 1 person 8, 1 person 7, 1 person 6, 1 person 5, 3 people 4 and 7 people have 2 videos.

## III. Summary of Relevant Studies

1.Automatic Deceit Detection Through Multimodal Analysis of High-Stake Court-Trials: In this pioneering study published in IEEE Transactions on Affective Computing, the authors utilize a novel multimodal framework that integrates video, audio, and physiological data for deceit detection. Their methodology involves the synchronization of facial expression analysis, linguistic processing of spoken words, and measurement of physiological responses such as heart rate and skin conductance to identify deceptive behavior in court-trial videos. Advanced machine learning models, including deep neural networks, are employed to analyze these complex datasets, resulting in a robust system capable of high-accuracy deception predictions.

2. Personality-Aware Deception Detection from Behavioral Cues: In his 2021 Master's thesis, Mandıra introduces an innovative personality-aware deception detection model that incorporates psychological profiling to evaluate behavioral cues in the context of individual personality traits. By first assessing the personality dimensions of subjects using established psychological tools, Mandıra then utilizes machine learning techniques to analyze how these personality traits influence deceptive behavior in controlled experimental settings. This approach provides a nuanced understanding of deception, recognizing the variability in deceptive tactics that are influenced by individual personality differences, thereby enhancing the precision and effectiveness of deception detection methods.

3.Deception Detection by 2D-to-3D Face Reconstruction from Videos: This research group focuses on the application of 2D-to-3D face reconstruction technologies to improve the visual analysis of facial cues for deception detection. They utilize advanced computer vision techniques and deep learning algorithms to transform 2D video frames into 3D facial models, which provide a richer dataset for analyzing subtle facial movements and expressions often linked with deceptive behavior. The 3D models allow for a more detailed and angle-invariant examination of micro-expressions, enhancing the accuracy of their detection system.

4.Multimodal Deception Detection Using Real-Life Trial Data: In their 2022 study, Sen et al. employ a multimodal approach to deception detection that effectively combines verbal content analysis with non-verbal behavior assessments derived from real-life trial videos. Utilizing natural language processing (NLP) tools, the study evaluates the speech content for linguistic cues of deception, while simultaneously employing machine learning techniques to analyze the synchronization and congruence between verbal and non-verbal cues, such as facial expressions and body gestures. This integration of multiple data sources enhances the accuracy of the deception detection system, showcasing the effectiveness of a comprehensive approach that captures a broader spectrum of deceptive behavior indicators.

5. Multi-Modal Deception Detection from Videos: In the realm of deception detection, Şen's doctoral dissertation (2020) stands out for its innovative approach to real-time integration of audio-visual cues, employing dynamic modeling to adapt to the temporal dynamics of deceptive behavior. This method continuously analyzes facial expressions and voice modulations, utilizing advanced machine learning algorithms and dynamic

models like Hidden Markov Models to track and interpret subtle behavioral changes over time. The framework's ability to adjust to different interaction contexts and continuously update its parameters enhances its effectiveness and accuracy in identifying deceptive cues, making it a significant contribution to the field.

## IV. Design justifications and explanations

In our project, we chose to explore two distinct solutions. The first approach involves "3D Convolutional Neural Network for Frame and Facial Analysis," while the second is centered around "Py-Feat-based Analysis and LSTM Application." Consequently, we have organized the "Theoretical Design," "Experiments and Results," and "Future Work" sections of our report into two separate parts to correspond with each approach.

### *A. 3D Convolutional Neural Network Approach to Frame and Facial Analysis*

We decided to obtain visual features from the 3D Convolutional Neural Network by processing sequences of facial expressions and movements across frames, which allows us to capture both spatial and temporal dimensions inherent in video data. The choice of utilizing a 3D Convolutional Neural Network for video analysis in deception detection is highly common due to the inherent nature of video as a three-dimensional data source (involving width, height, and time). Unlike 2D CNNs that process static images, 3D CNNs are capable of capturing temporal dynamics, which is crucial for analyzing sequences of facial expressions and movements that unfold over time[2]. This enables the network to detect temporal patterns and changes in behavior that may indicate deception. Moreover, 3D CNNs can handle both space and time information at once, giving a complete view of where things are and how they move in video frames. This is crucial for correctly spotting deceptive behavior in videos, as it helps capture both the arrangement of elements and their changes over time. This approach ensures a more robust analysis, making 3D CNNs an ideal choice for the complexities involved in deception detection from videos.

We decided to use only faces extracted from video frames to fine-tune our 3D Convolutional Neural Network model. This choice is based on the understanding that facial expressions are key in showing non-verbal signals, which are often signs of deception By isolating faces from the rest of the frame, we significantly reduce the noise and irrelevant background details that could detract from the model's performance. Focusing on facial features allows the network to specialize in detecting subtle changes in expressions that are critical for identifying deceptive behavior. This targeted approach not only improves the accuracy of deception detection but also enhances the efficiency of the training process, as the model learns from the most relevant and informative parts of the video data

In our deception detection project, we chose to use the 3D ResNet-50 model[6], which is widely known for its robust performance and accessibility. This model is based on deep residual learning, which makes it easier to train deep networks by handling the vanishing gradient problem through shortcut connections. These features are essential for processing the complex spatial and temporal data found in video sequences. Additionally, the widespread use of ResNet-50 in various applications has not only proven its effectiveness across multiple domains but also ensured that the model remains highly accessible for research and practical applications. Importantly, pretrained versions of 3D ResNet-50 are readily available, typically trained on large-scale video datasets such as Kinetics 400. These pretrained models offer a significant advantage by providing a strong foundational knowledge base, reducing the need for extensive training from scratch and enabling more efficient adaptation to specific tasks like deception detection. This combination of proven performance, accessibility, and the availability of pretrained models makes 3D ResNet-50 an ideal choice for our project.

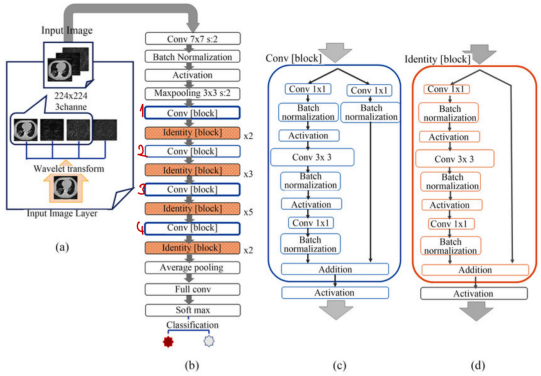


Fig. 1: 3D ResNet-50 Architecture[7]

Given that our dataset is sourced from 'YouTube' videos, we face significant variability in video quality and length. Additionally, the videos are inherently noisy, often featuring multiple subjects, with the main subject (the defendant) not consistently present in all frames. These factors complicate the preprocessing stage, making it challenging to prepare the data for effective training. We will discuss the detailed preprocessing steps in the experiments section of this report. After preprocessing, our dataset consists of 3D numpy arrays, each representing a sequence of facial frames extracted from the videos. These sequences are structured as 3D images, where each "depth" dimension corresponds to a consecutive facial frame. This format is particularly suited for input into our 3D ResNet-50 model.

As mentioned, we have chosen the 3D ResNet-50, pretrained on the Kinetics-400 dataset, as our model of choice. We utilize this pretrained model due to the limited availability of suitable data and the complexity of training such a model from scratch,

which helps mitigate the risk of overfitting. For our fine-tuning strategy, we plan to begin adjustments starting from the fourth block, as detailed in Figure 1. Typically, fine-tuning a model mostly involves modifying only the last fully connected layer; however, due to the specific nuances and variability in facial expressions related to deception in our dataset, deeper adjustments can be necessary. By extending fine-tuning to include an additional block rather than limiting it to the last layer, we aim to refine the model's ability to discern more difficult features that are important for accurate deception detection.

```
ResNetBasicHead(
  (pool): AvgPool3d(kernel_size=(8, 7, 7), stride=(1, 1, 1), padding=(0, 0, 0))
  (dropout): Dropout(p=0.5, inplace=False)
  (proj): Linear(in_features=2048, out_features=400, bias=True)
  (output_pool): AdaptiveAvgPool3d(output_size=1)
)
```
(1)

```
ResNetBasicHead(
  (pool): AvgPool3d(kernel_size=(8, 7, 7), stride=(1, 1, 1), padding=(0, 0, 0))
  (dropout): Dropout(p=0.5, inplace=False)
  (proj): Linear(in_features=2048, out_features=2, bias=True)
  (output_pool): Identity()
)
```
(2)

Fig. 2: Modifications on classifier block. (1) represents the original 3D ResNet, (2) represents our modifications.

### *B. Py-Feat-based Extraction of Visual Cues*

The Python Facial Expression Analysis Toolbox (Py-Feat) is a free, open-source package designed to enhance the analysis of facial expression data [Cheong et al., 2023]. Similar to tools like OpenFace, Py-Feat offers robust capabilities for extracting facial features but extends its utility with additional modules for preprocessing, analyzing, and visualizing facial expressions (Figure 3).

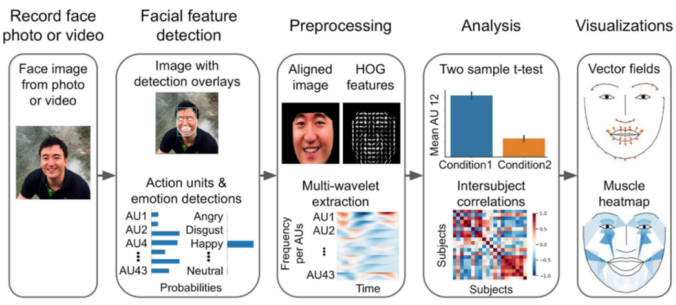


Fig. 3: Py-Feat Pipeline

Chosen detector set-up with Py-Feat is given in Figure 4

```
from feat import Detector

detector = Detector(
    face_model="retinaface",
    landmark_model="mobilefacenet",
    au_model="xgb",
    emotion_model="resmasknet",
    facepose_model="img2pose",
)

detector
```

Fig. 4: Py-Feat Detector Setup

Face detection in Py-Feat incorporates state-of-the-art CNNs such as the Multi-task Cascaded Convolutional Networks (MTCNN) and RetinaFace [Deng et al., 2019] for robust face detection. These models achieve high precision by performing simultaneous localization and facial landmark detection across varied conditions, including occlusions and diverse lighting environments.

Upon successful face detection, Py-Feat employs specialized CNN architectures to identify facial landmarks. These architectures, which may include adaptations of MobileNet [Howard et al., 2017] for enhanced speed and efficiency, are trained on extensive datasets to pinpoint critical facial regions with high accuracy. The landmark detection stage is crucial for subsequent analyses, as it establishes the geometric basis for more detailed expression analysis.

For detailed facial muscle movement analysis, Py-Feat utilizes action unit (AU) detection models that interpret facial landmarks to determine muscle activations. This process involves extracting geometric and appearance-based features, such as Histogram of Oriented Gradients (HOG), and classifying these features using advanced machine learning techniques like Support Vector Machines (SVM) and gradient boosting machines (XGBoost). For our study, XGBoost is being used for the detector due to trial and error of different models that we can use in Py-Feat framework.

Emotion recognition in Py-Feat is powered by deep residual networks enhanced with attention mechanisms, exemplified by the Residual Masking Network. These deep networks are trained on labeled datasets to classify facial expressions into discrete emotion categories, such as happiness, sadness, and anger.

## V. Experiments and Results

### *A. 3D Convolutional Neural Network Approach to Frame and Facial Analysis*

*1) Data Preprocessing:* Our dataset comprises 61 deceptive and 60 truthful trial videos, with lengths ranging from 4 to 60 seconds. The frame rates of these videos also vary, ranging from 10 to 30 frames per second. We had to exclude 5 videos from the dataset due to the absence of faces or the presence of occluded faces, leaving us with a total of 116 videos as suggested in. However, the remaining videos still present significant challenges due to their noisy nature. Some videos feature more than one subject, and the main subject (the defendant) is not always visible in all frames. This variability in video quality and content introduces additional complexities in the data processing stage.

After excluding 5 videos from our dataset, we removed 20% of the remaining videos as a test set. These test videos have not yet undergone the preprocessing steps that will be applied to the training set. Then, we segmented each video into subvideos

of 1.5 seconds in length, except for the final segments, which may be shorter due to their position at the end of the videos. For each subvideo, we consistently extracted evenly spaced 30 frames, regardless of their original frames per second (FPS), duration, and size. This standardization is important to ensure uniformity across the data, helping more effective learning and generalization by the neural network. After completing these operations, there are 1714 subvideos for training, each containing 30 frames.

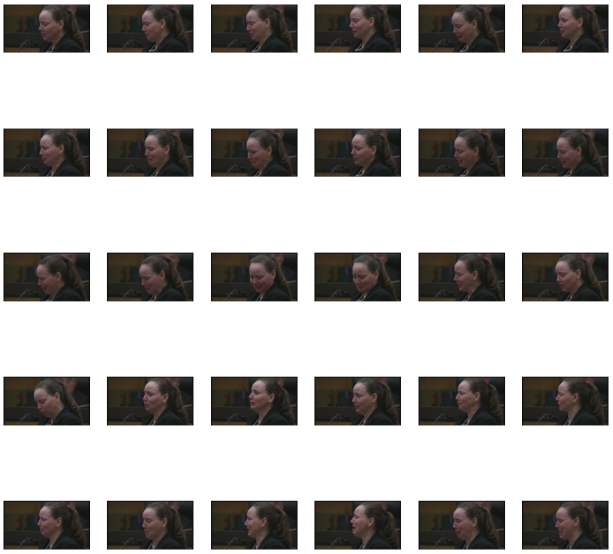

Fig. 5: Evenly spaced 30 frames obtained from a subvideo which has 1.5 seconds duration.

After segmenting the videos into subvideos and extracting 30 frames from each, we proceeded to isolate the defendant's face from every frame. For this task, we employed the CascadeClassifier function from the OpenCV library[11]. This model is designed for efficient object detection within images, utilizing simple patterns and rapid calculations to focus on likely areas while disregarding irrelevant regions. It leverages AdaBoost. Initially developed for swift face detection, the CascadeClassifier is particularly well-suited for real-time applications, offering an balance between high performance and speedy processing.

While the CascadeClassifier from the OpenCV library was initially used for face extraction due to its speed and simplicity, its performance was often unsatisfactory. The model sometimes failed to detect faces or extracted irrelevant objects. As a result, a more reliable deep learning-based model will be used to improve the accuracy and consistency of face extraction in the final analysis.

To solve the problems with the CascadeClassifier, we chose the RetinaFace[9] model for face extraction. RetinaFace is a state of art face detector that works well in different conditions as mentioned in Visual Cues part. In order to use the RetinaFace model for extracting faces from each frame, we utilized the DeepFace library developed by Serengil[13]. It uses a strong

Fig. 6: Evenly spaced 30 facial frames obtained from a subvideo by using CascadeClassifier

network with extra modules to better detect faces. The model predicts face locations and key points at the same time, making it very accurate. Its ability to handle different face sizes and angles makes it perfect for our project, giving us better and more reliable results. However, this model requires much more time and computational resources. Despite this, the improvement in accuracy justifies the additional cost.

Fig. 7: Evenly spaced 30 facial frames obtained from a subvideo by using RetinaFace

We present two figures to illustrate the differences between the CascadeClassifier and RetinaFace models. The first figure, using CascadeClassifier, shows many frames with no faces

or irrelevant parts extracted, demonstrating its inconsistent performance. In contrast, the second figure, using RetinaFace, displays 30 correctly extracted face frames, proving its accuracy compared to CascadeClassifier. Accurate face extraction is necessary for our training process, as it ensures the quality and consistency of the input data, which results to better model performance. The significant difference between the two figures emphasizes the importance of using a reliable model like RetinaFace.

Following the extraction and face detection processes, we compiled the 30 frames from each subvideo into a sequence to create a 3D image representation of the subvideo. Each frame within this sequence acts as a distinct temporal layer, contributing to the depth of the 3D image. This depth is essential because it covers the progression of events over time, providing a dynamic and comprehensive view of each segment. Consequently, this structured format of data is ideally suited for analysis via 3D CNNs, allowing the model to effectively capture both spatial features and temporal dynamics, which is critical for accurate deception detection. This method ensures that our dataset is optimally prepared to increase the sophisticated capabilities of 3D convolutional neural networks.

Each subvideo is represented as a 3D image, constructed from a 4D matrix consisting of channel, depth (representing discrete time points), height, and width. To make these data work with PyTorch, we combined 1714 of these matrices into numpy arrays, creating a dataset that is organized as 5D matrices. The dimensions of these matrices are arranged in the following order: image number, channel, depth, width, and height. This structured arrangement is important for systematic processing and analysis. Subsequently, we converted these 5D matrices into the torch Dataset format. This conversion is important since it enables the dataset to be integrated into the PyTorch framework, making it easier to fine-tune the CNN model. This preparation ensures that our dataset is fully compatible with deep learning workflows, optimizing it for better performance during model training. At the end, data is normalized with values obtained from Kinetics-400 dataset. Kinetics dataset normalization generally uses mean and standard deviation values computed from the dataset. Common values used are:

- Mean: [0.43216, 0.394666, 0.37645]
- Standard Deviation: [0.22803, 0.22145, 0.216989]

*2) Fine Tuning 3D ResNet50:* In the theoretical design section, we outlined our planned approach for fine-tuning the 3D ResNet-50 model, which is pretrained on the Kinetics-400 dataset. As part of this process, we imported the pretrained model and then selectively froze the weights in certain blocks of the model, as previously discussed. This was done to prevent the calculation of gradients and to ensure that the weights in these blocks remain unchanged during training. By doing so, we can focus on fine-tuning only the specific blocks that are crucial for adapting the model to our specific task. With these preparations in place, the model is now ready to undergo the fine-tuning procedure, allowing us to optimize performance for deception detection in video data.

As previously mentioned, our goal is to fine-tune the 3D ResNet-50 model, which was pretrained on the Kinetics-400 dataset. Initially, we organized our data into batches, each containing 8 items. However, during the initial attempt at forward propagation with a sample, we encountered an error. This issue arose because the original height and width dimensions of our images, 128x128, were incompatible with the model. To resolve this, we resized the height and width dimensions to 224x224, matching the dimensions used in the Kinetics-400 dataset training. Despite this adjustment, we faced several numpy errors while attempting to create a dataset with the resized facial frame images.

To adapt our dataset for use with the 3D ResNet model, we introduced zero padding at both the beginning and end of the time/frame dimension. This modification helps standardize video lengths across the dataset, ensuring they fit well into our neural network architecture. We upgraded our facial feature extraction to RetinaFace, which proved far superior to the older CascadeClassifier technique. This upgrade significantly cut down on the need for complex preprocessing, like advanced face normalization, though we continued to implement some fundamental preprocessing to maintain uniformity across all frames.

Working with 3D images introduced unique challenges, particularly in terms of data management and processing speed. These included the need for increased memory capacity and enhanced processing capabilities to handle the volume and complexity of the data efficiently. To tackle these challenges, we utilized an NVIDIA A100 GPU with ample RAM from Google Colab, providing the high computational power and memory needed to effectively manage and train our sophisticated 3D datasets.

*Training and Hyperparameter Search:* As mentioned earlier, we divided our dataset into training and testing sets right at the start of preprocessing. After turning the 1.5-second subvideos from each video into 3D images, we ended up with an equal number of 3D images and subvideos. Before we began training, we split the training dataset again, setting aside 20% as a validation set to help us fine-tune the model settings. During training, we don't keep track of which original video each subvideo comes from. Instead, we treat each subvideo as a separate item. We give each 3D image a label based on whether the original video it came from was deceptive or not. This method ensures that our model treats each piece of video independently, which helps it perform better across different types of videos.

During training, after several trials, we decided to implement a significant weight decay of 0.01 to combat overfitting. This approach was necessary because the model quickly learned the critical features for detecting deception, which increased the risk of overfitting. To improve our model's settings, we trained ten different versions, adjust hyperparameters such as the optimizer, learning rate, and learning rate scheduler to find the optimal configuration. The train-validation loss figures, which are shown below, helped us identify the most effective model settings.

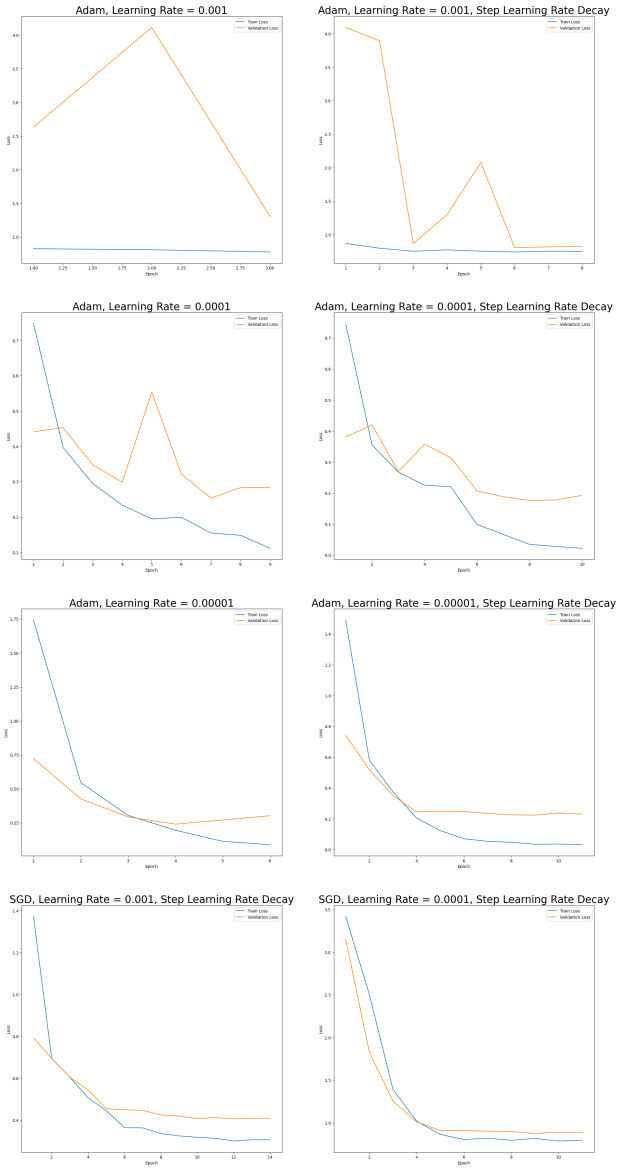


Fig. 8: Train - Validation vs Epoch plots of 10 different configurations

The results aligned with our expectations, indicating that Adam outperforms SGD, especially at lower learning rates. Notably, Adam with a learning rate of 0.001 and SGD at 0.0001 both yielded suboptimal outcomes. Given Adam's consistent superiority in performance over SGD, we expanded our exploration with additional configurations using Adam. Interestingly, we observed that implementing a step learning rate decay did not positively influence the outcomes, suggesting that the benefits of learning rate decay might be context-dependent and require further investigation. However, fine-tuning the scheduler's configuration needs more detailed exploration because we used one specific configuration and do not discover the optimal parameters. The optimal performance was achieved using Adam with a learning rate of 0.001, without employing step learning rate decay, according to our comparative analysis of the train-validation loss figures.

A notable observation from the figure is that, particularly in the initial epochs, the test loss can be lower than the train loss. This phenomenon was attributed to the pretrained 3D ResNet's rapid learning capabilities. During the early training phase, the model parameters are updated batch by batch, causing the training loss to drop significantly. By approximately halfway through the first epoch, the training loss sharply decreases from 3.5 to 0.5. After the completion of the first epoch, the validation process takes place, using the parameters updated during that epoch. As the reported loss values are averages across all batches within the epoch, this difference leads to what might appear as misinformation. Below, we provide the loss values for the first epoch of our best-performing model:

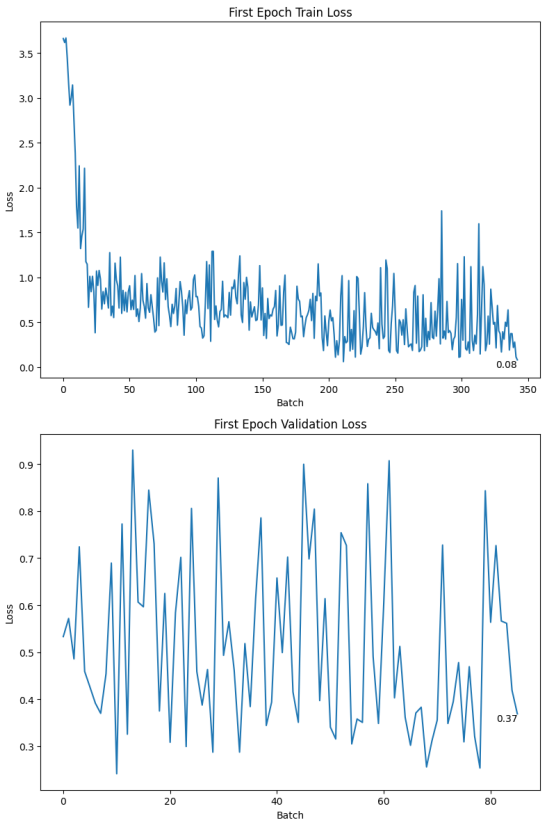


Fig. 9: Train and validation Loss with respect to batches at first epoch for best model

*3) Testing:* For testing, we employed a strategy different from that used in training. As previously detailed, the train and test datasets were separated at the project's outset. Specifically, 12 videos for each label were seperated for testing. These videos were preprocessed in the same way as those in the training set. However, during testing, subvideos from each main video are not treated as independent units. Instead, each subvideo, now represented as a 3D image, is processed by the model which only accepts 3D inputs. To evaluate the videos as whole units rather than individual subvideos, we track which major video each subvideo originates from. We compute the logit for each subvideo, average these logits for each video, and then base our final deception prediction on this average logit value.

We used the top two performing models to obtain the training results. The outcomes are presented below:

TABLE I: Fine-Tuned 3D ResNet Test Results

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Best Model | 0.863 | 0.9 | 0.818 | 0.818 |
| Second Model | 0.818 | 1 | 0.636 | 0.778 |

The optimal model utilized an Adam optimizer with a learning rate of 0.0001 and no learning rate scheduler, while the secondary model employed an Adam optimizer at the same learning rate but included a step learning rate decay. The performance metrics achieved by these models were exceptionally high, which could suggest either a limited number of test samples or potential data leakage. Careful examination of these factors is essential to ensure the robustness and reliability of our results.

### *B. Py-Feat-based Analysis*

*1) Data Preprocessing with Py-Feat:* After the before mentioned configured detector is run for all the videos in the dataset, an excel file is being created for each video. For this task, we configured this detector to skip every 30 frame to analyze the results. Since we will use Action Units and Emotions for our task, only these 2 parameters will be displayed. The results for 2 skipped frame is given in Figure 10.

The Facial Action Coding System (FACS) is a scientific tool designed to classify every conceivable human facial expression by breaking them down into individual components called AUs. Initially developed by Carl-Herman Hjortsö in 1978 and expanded by Paul Ekman and Wallace V. Friesen in 2002, FACS categorizes facial expressions based on the underlying muscle movements. Each AU corresponds to specific muscle actions that produce changes in facial appearance, such as the raising of an eyebrow or the tightening of lips.

As mentioned before, we use the Py-Feat library with its default multiperson detection model to obtain 18 binary indicators of Action Units (AUs) for each frame in our videos. These include: AU1 (inner brow raiser), AU2 (outer brow raiser), AU4 (brow lowerer), AU5 (upper lid raiser), AU6 (cheek raiser), AU7 (eyelid tightener), AU9 (nose wrinkler), AU10 (upper lip raiser), AU12 (lip corner puller), AU14 (dimpler), AU15 (lip corner depressor), AU17 (chin raiser), AU20 (lip stretcher), AU23 (lip tightener), AU25 (lips part), AU26 (jaw drop), AU28 (lip suck), and AU45 (blink). We average these binary indicators through the frames to obtain a single AU feature for each video.

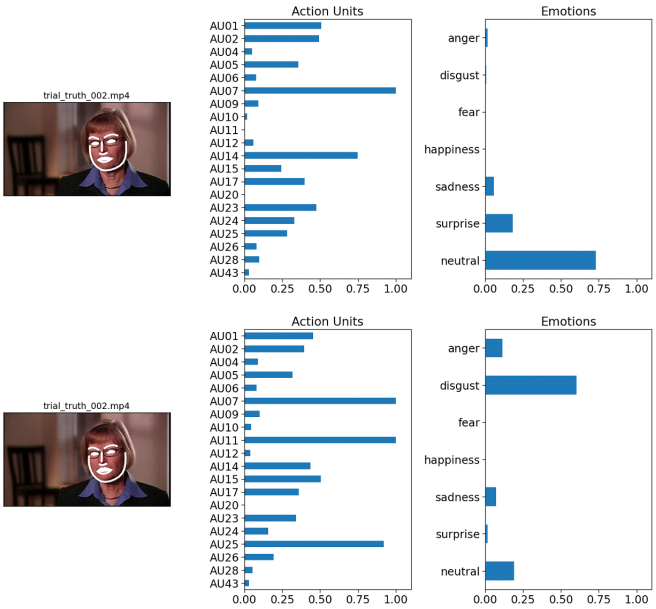


Fig. 10: Action Unit and Emotion Output for 2 Different Frames

Also, it is possible to plot the emotion distribution throughout the each video as in Figure 11 which can be quite beneficial to take into account while trying to classify deception.

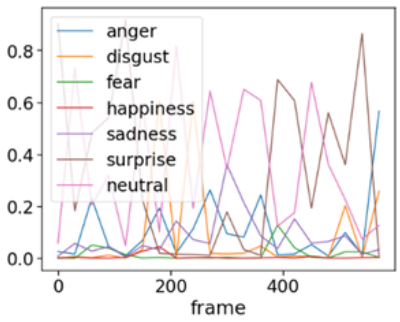


Fig. 11: Emotion plot for one video

*2) Dataset Preparing for Models:* The dataset consists of 121 trial videos of which 61 are labeled as deceptive and 60 are labeled as true. In this dataset, some data-instances belong to the same person and same trial. 51 people have only one video. The distribution of others as follows: 1 person 18 videos, 1 person 8, 1 person 7, 1 person 6, 1 person 5, 3 people 4 and 7 people have 2 videos. Train dataset will consist of unique trials with unique persons to prevent the model to learn the same person's behaviour more than one time. This may reduce overfitting and help model to generalize better overall.

*a) Data Extraction:* We employed Py-Feat to extract Action Units (AUs) and Emotions from the videos in the RLT dataset. These features were crucial for detecting deceptions and were used as inputs for our models. The extraction process ensured that the relevant visual cues were accurately captured and represented in a format suitable for model training.

*b) Leave-One-Person-Out (LOPO) Validation:* To handle the imbalanced distribution of videos per subject, we utilized the Leave-one-person-out (LOPO) validation method. This approach involved creating 53 folds from 116 videos, ensuring no overlap of subjects between the training and test sets. Subjects with very few (1) or an excessive number of videos (20% of the remaining videos) were always included in the training set. From the remaining videos, 15% to 20% were randomly selected for the validation set. To balance the training set, we downsampled the majority class, resulting in an equal number of instances from each class.

*3) Model Training Approaches:*

*a) Person-Aware Models:* For person-aware models, we categorized each video based on the subject. All videos for each person were grouped, and the extracted features (Action Units and Emotions) were treated as a single unit per subject. This ensured that the training and validation sets did not have overlapping subjects. We applied the LOPO validation method to these grouped outputs to train our model effectively, leveraging the subject-specific data.

*b) Person-Unaware Models:* In contrast, person-unaware models were trained on data without considering the subject identity. The extracted features were used to train these models randomly, treating each video independently. This approach allowed us to evaluate the model's performance without the influence of subject-specific features.

By employing these two distinct approaches, we could compare the impact of subject awareness on the model's ability to detect deception, providing a comprehensive analysis of the dataset and the efficacy of our models.

*4) Training and Evaluation:* We used PyTorch for implementing our deep learning models. The visual GRU model was trained for a maximum of 20 epochs with the Adam optimizer, a learning rate of 1e-3, a weight decay of 1e-6, and different batch sizes of 4, 16, and 32 were tried. The GRU model was trained for 20 epochs.

Both person-unaware and person-aware models were trained and evaluated. The performance of the models was assessed using the following metrics: accuracy, precision, recall, and F1-score.

*5) Model Selection:* For the task of deception detection, we selected Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models due to their proven effectiveness in handling sequential data and capturing long-term dependencies. These models were used separately to compare their performance in this specific application. Reasoning for choosing these models are explained below.

Long Short-Term Memory (LSTM): LSTM networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies. They are designed to remember information for long periods, making them suitable for tasks where the context over sequences is important. This is particularly relevant in deception detection, where temporal patterns in facial expressions and emotions can be crucial indicators of deceptive behavior. LSTMs use memory cells and gating mechanisms to control the flow of information, thus effectively addressing the vanishing gradient problem common in traditional RNNs.

Gated Recurrent Unit (GRU): GRUs are a simplified version of LSTMs that also address the vanishing gradient problem. GRUs combine the input and forget gates into a single update gate and merge the cell state and hidden state, resulting in a more streamlined architecture. This simplicity often leads to faster training times while maintaining performance comparable to LSTMs. GRUs are particularly effective in scenarios where computational efficiency is critical, and they perform well in capturing temporal dependencies in sequential data.

*a) Expected Performance in Deception Detection:* Both LSTM and GRU models were evaluated separately to determine their effectiveness in the deception detection task due to the following reasons:

Temporal Dynamics: Deception detection relies on the ability to capture and analyze changes in facial expressions and emotions over time. Both LSTM and GRU models are well-suited for this as they can effectively model the temporal dynamics and dependencies within the video sequences.

Handling Sequential Data: The sequential nature of video data, where each frame is dependent on the previous frames, makes LSTM and GRU models ideal. These models can maintain a memory of past frames and use this information to make predictions about deception in the current frame.

Robustness to Noise: Real-world video data can be noisy and contain irrelevant information. LSTM and GRU models, with their gating mechanisms, can selectively retain useful information and forget irrelevant details, making them robust to noise and capable of focusing on significant features.

Efficiency: While LSTMs are powerful, their complexity can lead to longer training times. GRUs, being a simpler alternative, offer a balance between performance and efficiency, making them a practical choice for large datasets or when computational resources are limited.

Comparison of LSTM and GRU Models: By leveraging the strengths of both LSTM and GRU models, we conducted a comparative analysis to determine which model performs better in the context of deception detection. We trained and evaluated both models separately, using the same dataset and experimental setup, to provide a fair comparison. The performance of

each model was assessed based on metrics such as accuracy, precision, recall, and F1-score.

This comparative approach allowed us to understand the strengths and weaknesses of each model in handling the temporal patterns and noise inherent in the video data, ultimately guiding the selection of the most suitable model for our deception detection system.

*6) Reevaluation of Previous Methods:* To ensure the reliability of our results, we compared our findings with previous studies. Specifically, we observed that some previously reported results might have discrepancies due to differences in implementation details and validation methodologies.

For instance, Sen et al. (2022) reported accuracy results of 76.27% and 77.40% for their visual model using Support Vector Machine (SVM) and Random Forest (RF) classifiers, respectively. Additionally, they reported an accuracy of 80.79% using a Neural Network (NN) classifier. While Sen et al. used Leave-One-Person-Out (LOPO) validation, we conducted our reevaluation to ensure that the same rigorous validation approach was applied consistently and to verify the robustness of the reported metrics under our experimental conditions.

Each SVM, RF, and NN result was obtained by taking an average of 20 repetitions, ensuring robustness in the reported metrics. The reevaluation with our LOPO validation approach is crucial for a fair comparison and understanding of true model capabilities in a real-world scenario.

*7) LSTM Model Application:* To ensure the integrity and uniformity of our dataset, missing values were filled with the mean values of respective features, maintaining the statistical properties essential for robust model training. We standardized the dataset using the DeceptionDataset class, ensuring a consistent input format across varying frame counts. Specifically, each video was standardized to a fixed number of frames (30 frames), crucial for reliable processing of video data.

Data loaders were configured with a batch size of 4 to optimize computational efficiency while effectively learning from the dataset. Shuffling was enabled in the training dataset to minimize bias and enhance the learning process, while it was disabled in the test dataset to ensure consistent evaluation conditions.

The LSTM model was specifically designed with 68 input features, 128 hidden units, and two layers to adeptly manage the temporal dependencies inherent in video data, crucial for distinguishing between truthful and deceptive behaviors. We chose binary cross-entropy loss and the Adam optimizer for their effectiveness in managing sparse gradients and adapting learning rates, crucial for minimizing loss over the training period. To increase the model's robustness and prevent overfitting, we integrated dropout layers with a probability of 0.5 and L2 regularization with a weight decay parameter of 0.01.

During the initial training phases, the model exhibited a gradual reduction in average loss from 0.709 in the first epoch to 0.663 in the tenth epoch, demonstrating effective learning. However, further optimization is needed to improve the model's performance, given the higher-than-ideal loss values.

After training, the model was evaluated on a separate test dataset and achieved an accuracy of 64 percent, indicating a moderate capability to differentiate between deceptive and truthful outcomes. This suggests that while the model has learned to some extent, there is considerable scope for optimization.

*8) GRU Model Application:* To ensure the integrity and uniformity of our dataset, missing values were filled with the mean values of respective features, maintaining the statistical properties essential for robust model training. We standardized the dataset using the DeceptionDataset class, ensuring a consistent input format across varying frame counts. Specifically, each video was standardized to a fixed number of frames (30 frames), crucial for reliable processing of video data.

Data loaders were configured with a batch size of 4 to optimize computational efficiency while effectively learning from the dataset. Shuffling was enabled in the training dataset to minimize bias and enhance the learning process, while it was disabled in the test dataset to ensure consistent evaluation conditions.

The GRU model was designed with 68 input features, 64 hidden units, and three layers to adeptly capture temporal dependencies in video data. We employed binary cross-entropy loss and the Adam optimizer for managing sparse gradients and adapting learning rates during training. Additionally, dropout layers with a probability of 0.5 and L2 regularization with a weight decay parameter of 1e-6 were integrated to increase the model's robustness and prevent overfitting.

During training, the model exhibited a gradual reduction in average loss from 0.709 in the first epoch to 0.663 in the tenth epoch, indicating effective learning. However, further optimization is needed to improve the model's performance, given the higher-than-ideal loss values.

After training, the model was evaluated on a separate test dataset, achieving an average accuracy of 64 percent. While the model demonstrated a moderate capability to differentiate between deceptive and truthful behaviors, there is considerable room for improvement.

*9) Results and Discussion:*

*a) Model Comparison:* In this section, we present the performance of LSTM and GRU models for deception detection. We evaluated these models under two conditions: person-aware and person-unaware. The results are compared with previously reported results by Sen et al. (2022) using different classifiers.

*b) Person-Aware Deception Module:* The performance results for the person-aware deception module are presented in Table II. The LSTM model achieved an accuracy of 80.62%, a precision of 0.7940, a recall of 0.8071, and an F1-score of 0.8005. The GRU model demonstrated slightly lower performance with an accuracy of 79.37%, a precision of 0.7955, a recall of 0.7998, and an F1-score of 0.7976.

TABLE II: Results of the person-aware deception module

| Model | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| LSTM | 0.8062 | 0.7940 | 0.8071 | 0.8005 |
| GRU | 0.7937 | 0.7955 | 0.7998 | 0.7976 |

*c) Person-Unaware Deception Module:* The performance results for the person-unaware deception module are presented in Table III. The LSTM model achieved an accuracy of 74.66%, a precision of 0.7594, a recall of 0.7684, and an F1-score of 0.7639. The GRU model showed lower performance with an accuracy of 71.25%, a precision of 0.7310, a recall of 0.7282, and an F1-score of 0.7296.

TABLE III: Results of the person-unaware deception module

| Model | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| LSTM | 0.7466 | 0.7594 | 0.7684 | 0.7639 |
| GRU | 0.7125 | 0.7310 | 0.7282 | 0.7296 |

*d) Comparison with Previous Work:* Sen et al. (2022) reported accuracy results of 76.27% and 77.40% for their visual model using Support Vector Machine (SVM) and Random Forest (RF) classifiers, respectively. Additionally, they reported an accuracy of 80.79% using a Neural Network (NN) classifier.

Comparing our results with those reported by Sen et al. (2022), we observe the following:

- Our person-aware LSTM model outperforms their SVM and RF classifiers, with an accuracy of 80.62% compared to 76.27% and 77.40%, respectively. The performance is slightly lower than their NN classifier, which achieved 80.79%.
- Our person-aware GRU model also shows competitive performance with an accuracy of 79.37%, which is higher than their SVM and RF classifiers but slightly lower than their NN classifier.
- For the person-unaware models, both LSTM and GRU show lower accuracies (74.66% and 71.25%, respectively) compared to Sen et al.'s results. This highlights the importance of person-awareness in enhancing model performance for deception detection.

Overall, our LSTM and GRU models demonstrate strong performance in the person-aware setting, indicating the effectiveness of recurrent neural networks in capturing temporal patterns for deception detection. The comparison underscores the necessity of rigorous validation approaches and the potential benefits of model designs that account for subject-specific variations.

## VI. Main Results

In this work, we explored the impact of incorporating emotional features alongside Action Units (AUs) to enhance the performance of deception detection models. The novelty of our approach lies in the integration of emotions, aiming to leverage the rich, dynamic information they provide about a person's state during deceptive behavior.

### *A. Impact of Emotions on Deception Detection*

To evaluate the effect of adding emotions, we used the Py-Feat library to extract both AUs and emotional features from the video data. These features were then used to train both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models under person-aware and person-unaware settings.

For the person-aware models, the LSTM model achieved an accuracy of 80.62%, while the GRU model achieved an accuracy of 79.37%. Despite the inclusion of emotional features, the improvement over models using only AUs was marginal. The performance metrics for precision, recall, and F1-score also reflected this trend, showing slight but not significant enhancements.

In the person-unaware setting, the LSTM model achieved an accuracy of 74.66%, while the GRU model showed a lower accuracy of 71.25%. Similar to the person-aware setting, the addition of emotional features did not significantly boost the performance of the models. The precision, recall, and F1-score for these models were consistent with their accuracy results, indicating that the inclusion of emotions did not provide the expected performance gains.

### *B. Effect of Data Preprocessing for 3D ResNet*

Our dataset, after preprocessing, comprised 116 videos (61 deceptive and 55 truthful), divided into subvideos of 1.5 seconds each. This resulted in 1714 subvideos for training, each containing 30 frames. Initially, we used the CascadeClassifier function from the OpenCV library for face extraction. However, this approach resulted in many frames with missing or incorrectly identified faces. The poor performance of CascadeClassifier significantly reduced the model's efficiency, as the absence or incorrect identification of faces led to noisy and unreliable input data. To address this issue, we switched to the RetinaFace model, which greatly improved accuracy and consistency in face detection, ensuring high-quality face data for our 3D CNN model.

### *C. Benefits of Fine-Tuning a Pretrained Model*

The efficiency of fine-tuning the pretrained 3D ResNet-50 model was significantly enhanced due to several factors: reduced training time, improved generalization, mitigation of overfitting, and enhanced feature extraction. Leveraging a pretrained model allowed us to bypass the computationally expensive process of training from scratch, as the 3D ResNet-50 model had already learned to recognize a wide variety of features relevant to video analysis from the extensive Kinetics-400 dataset. By freezing certain layers, we focused on fine-tuning those most relevant to our specific task, which prevented overfitting and ensured the model maintained a balance between learning new task-specific features and retaining robust general features. This approach provided advanced feature extraction capabilities, capturing both spatial and temporal dimensions

inherent in video data, which was crucial for accurately analyzing sequences of facial expressions and movements to detect deception.

### *D. Comparison with Previous Work*

Our findings were compared with those reported by Sen et al. (2022), who achieved accuracy results of 76.27% and 77.40% using Support Vector Machine (SVM) and Random Forest (RF) classifiers, respectively, and 80.79% using a Neural Network (NN) classifier. While our person-aware LSTM model achieved a competitive accuracy of 80.62%, and the GRU model also performed well with 79.37%, it is evident that our models are on par with or slightly better than traditional classifiers. However, in the person-unaware setting, our models performed slightly lower, indicating the importance of person-specific information in deception detection. In comparison, our 3D ResNet-50 model achieved an accuracy of 86.3%, which is notably higher than the results reported by Sen et al. This significant improvement underscores the effectiveness of using a 3D convolutional neural network for capturing spatial and temporal features in video data. However, this high accuracy raises concerns about potential biases. The model may have benefited from overfitting due to the limited size of the dataset, leading it to learn specific patterns that do not generalize well to new data. Additionally, the preprocessing steps, such as the use of the RetinaFace model for face extraction, might have introduced biases by ensuring high-quality inputs that are not reflective of real-world conditions. The variability in video quality and the exclusion of certain challenging frames during preprocessing could also have contributed to inflated performance metrics. Therefore, while the results are promising, further validation with a larger and more diverse dataset is essential to confirm the robustness and generalizability of the 3D ResNet-50 model's performance in real-world deception detection scenarios.

### *E. Unexpected Findings*

Despite the initial hypothesis that adding emotional features would significantly enhance the detection of deception, our results indicate otherwise. The inclusion of emotions did not yield the expected increase in model performance. This suggests that while emotions provide valuable context, the AUs alone capture most of the necessary information for detecting deception in this dataset. This finding underscores the complexity of effectively integrating emotional data into deception detection models.

### *F. Future Work*

Given the unexpected findings, several avenues for future research are suggested. First, exploring more sophisticated methods for integrating emotional data could provide better insights. Techniques such as attention mechanisms or multimodal fusion strategies might be employed to more effectively combine AUs and emotional features. Additionally, investigating other types of emotional representations, such as continuous emotional states or valence-arousal models, might yield different results.

Moreover, expanding the dataset to include more diverse emotional expressions and deceptive scenarios could enhance the model's robustness and generalizability. Another promising direction is to explore transfer learning approaches, where models pretrained on large, diverse datasets can be fine-tuned for deception detection tasks.

Finally, incorporating other modalities, such as audio or physiological signals, could provide a richer context and improve detection accuracy. Multimodal approaches that combine visual, auditory, and physiological data might offer a more comprehensive understanding of deceptive behavior.

In conclusion, while our study highlights the potential of LSTM and GRU models in deception detection, the anticipated advantage of incorporating emotional features did not materialize. These findings contribute to the ongoing research in multimodal deception detection, emphasizing the need for continued exploration and innovation in integrating diverse data sources for improved performance.

One promising direction is to integrate the 3D Convolutional Neural Network (3D ConvNet) with sequential models such as Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRUs). This hybrid approach can leverage the spatial and temporal feature extraction capabilities of 3D ConvNets and the sequential modeling strengths of LSTMs or GRUs. By combining these models, we can better capture long-term dependencies and dynamic changes in facial expressions and movements, potentially improving the accuracy and reliability of deception detection.

To address the limitations of our dataset and reduce the risk of overfitting, data augmentation techniques can be employed. Augmentation methods such as random cropping, rotation, scaling, and horizontal flipping can create a more diverse set of training examples, enhancing the model's ability to generalize to new data. Additionally, synthetic data generation techniques, such as Generative Adversarial Networks (GANs), can be explored to produce realistic video data, further enriching the training dataset.

Incorporating additional modalities such as audio and physiological signals can provide a more comprehensive understanding of deceptive behavior. Multimodal fusion techniques that integrate visual, auditory, and physiological data can capture a wider range of deception cues, potentially improving detection accuracy. Exploring advanced fusion strategies, such as attention mechanisms or hierarchical fusion, can help effectively combine these diverse data sources.

Given the success of using pretrained models like 3D ResNet-50, further experimentation with transfer learning and fine-tuning on different large-scale video datasets can be beneficial. Fine-tuning models pretrained on diverse datasets such as Kinetics-600 or Sports-1M may provide additional performance gains by leveraging a broader range of learned features.

## VII. Group members and Responsibilities

Okan Ertürk - Data Preprocessing and Analysis with Py-Feat and Model Selection/Training with LSTM and GRU

Damla Erden - Data Preprocessing and Analysis with Py-Feat and Model Selection/Training with LSTM and GRU

Ali Yılmaz - 3D Convolutional Neural Network Approach to Frame and Facial Analysis

## VIII. References


[1] Biçer, B., & Dibeklioğlu, H. (2024). Automatic deceit detection through multimodal analysis of high-stake court-trials. IEEE Transactions on Affective Computing, 15(1), 342-356.

[2] Mandıra, B. (2021). Personality-Aware Deception Detection from Behavioral Cues [Master's thesis, Bilkent University].

[3] Ngo, M. L., Mandira, B., Yılmaz, S. F., Karaoglu, S., Bouma, H., Dibeklioğlu, H., Gevers, T., & Heij, W. (2018). Deception Detection by 2D-to-3D Face Reconstruction from Videos. arXiv:1812.10558v1 [cs.CV].

[4] Sen, M. U., Perez-Rosas, V., Yanikoglu, B., Abouelenien, M., Burzo, M., & Mihalcea, R. (2022). Multimodal Deception Detection Using Real-Life Trial Data. IEEE Transactions on Affective Computing, 13(1), 306-319.

[5] Şen, M. U. (2020). Multi-Modal Deception Detection from Videos [Doctoral dissertation, Sabancı University].

[6] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A Closer Look at Spatiotemporal Convolutions for Action Recognition. arXiv. https://arxiv.org/abs/1711.11248v3

[7] Matsuyama, E. (2020). A Deep Learning Interpretable Model for Novel Coronavirus Disease (COVID-19) Screening with Chest CT Images. Journal of Biomedical Science and Engineering, 13(7), 140-152. https://doi.org/10.4236/jbise.2020.137014

[8] Cheong, J. H., Xie, T., Byrne, S., Chang, L. J. (2023). Py-Feat: Python Facial Expression Analysis Toolbox. Affective Science, 4(4), 781–796. https://doi.org/10.1007/s42761-023-00191-4

[9] Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., Zafeiriou, S. (2019, May 2). RetinaFace: Single-stage dense face localisation in the wild. arXiv.org. https://arxiv.org/abs/1905.00641

[10] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017, April 17). MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv.org. https://arxiv.org/abs/1704.04861

[11] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1, I-I. https://doi.org/10.1109/CVPR.2001.990517

[12] Ngô, L. M., Wang, W., Mandira, B., Karaoğlu, S., Bouma, H., Dibeklioğlu, H., & Gevers, T. (2021). Identity unbiased deception detection by 2D-to-3D face reconstruction. In Proceedings of the Winter Conference on Applications of Computer Vision (WACV).

[13] Serengil, S. I. (n.d.). DeepFace: A lightweight face recognition and facial attribute analysis library for Python [Source code]. GitHub. Retrieved May 16, 2024, from https://github.com/serengil/deepface