Machine Learning Approach for Spontaneous Facial Expression Recognition

Maxine Annel Pacheco Ramírez
Department of Electrical Engineering,
University of Houston, TX, USA
mpachec4@cougarnet.uh.edu

Aayush Gupta
Department of Computer Science
University of Houston, TX, USA
agupta56@cougarnet.uh.edu

Abstract

This study outlines a comprehensive approach for identifying spontaneous facial expressions using basic machine learning techniques and advanced methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The research utilizes the Denver Intensity of Spontaneous Facial Action (DISFA) dataset, which features high-resolution videos annotated with the intensity of 12 distinct Action Units (AUs) rated from 0 to 5, based on the Facial Action Coding System (FACS). Initially, the project implements traditional techniques such as Support Vector Machine (SVM), Random Forest (RF), and K-nearest Neighbor (KNN). Subsequently, it incorporates CNNs and RNNs to analyze temporal patterns, thus enabling the interpretation of dynamic expressions over time. Our objective is to enhance the accuracy of baseline models through advanced feature extraction and machine learning techniques to provide insights into the practical uses of such technologies in fields like security, healthcare, and interpersonal communication. The findings from this study suggest that Random Forest and KNN overall perform better than advanced Deep Learning Techniques.

1 Introduction

Faces and facial expressions play a significant role in engaging and interacting with others in social contexts (Salama AbdELminaam et al., 2020). As it is known, facial behavior provides important information about a person's emotions and intentions (Liu et al., 2014). Currently, this area of research is attracting interest from a wide range of fields, since it has the potential to benefit several fields, including medicine, behavioral science, communication, education, business, and security, among others(Yin et al., 2006) (Barnouti et al., 2016). Due to its broad applications several methods have been developed to extract and recognize facial expressions within the domain of machine learning and computer science (Zeng et al., 2007).

The purpose of this study is to make use of a dataset containing well-labeled data on the topic of recognizing spontaneous facial expressions. According to the background research performed, more descriptive approaches should be utilized for developing models regarding this topic, in which the configuration of the facial action is described without reference to its meaning. An example is the FACS Action Units (AUs), which are anatomically based on facial actions. Each Action Unit (AU) is based on a particular facial muscle. In several fields of study, the use of the Facial Action Coding System (FACS), which is used to describe facial movements, has been used to examine emotions, pain, or psychological conditions(Cohn et al., 2007) (Rosenberg and Ekman, 2020). By using this metric, the intensity of the AUs may be graded based on changes in intensity (Ekman and Friesen, 1978).

The database we have selected for our project's development is DISFA. This choice arises from its alignment with our research objectives, as it provides access to well-structured and labeled information. DISFA database meets the criteria necessary for model creation, which includes 1) The application of the FACS approach, and 2) The capability to work with the measurement of Action Units (AUs) in facial expressions. Our plan for the project starts with the proper processing of videos into images and calculating the features for each video frame, as part of our research, we propose the inclusion of additional features to see how they may influence the model, as well as to explore further adjustments to beat the baseline such as comparing various machine learning techniques and fine-tuning the model to optimize it.

2 Data Description

The DISFA dataset comprises high-resolution videos of 26 participants (11 females and 15 males) from various ethnic backgrounds. These participants varied in age from 18 to 50 years and consisted of three Asians, 21 Euro-Americans, two Hispanic, and one African-American, participants were showing spontaneous facial expressions while watching emotionally charged videos.

Participants viewed a 4-minute video clip (242 seconds in length) intended to elicit spontaneous AUs in response to videos that elicit a range of facial expressions of emotion. While viewing the video, participants sat in a comfortable chair positioned in front of a video display and stereo cameras. They were alone with no one else present, their facial behavior was imaged using a high-resolution (1024 × 768 pixels) BumbleBee Point Grey stereo-vision system at 20 fps under uniform illumination. For each participant, there are 4845 video frames (Mavadati et al., 2013), and each video frame shows the presence and intensity (on a 0-5 scale) of 12 Action Units, providing a comprehensive framework for analyzing facial behavior.

3 Related Work and Approach

In the context of prior research on the application of Action Units (AUs) for facial behavior recognition, we found several databases using the Facial Action Coding System (FACS). Literature highlights two extensively utilized databases: the "Cohn-Kanade Expression Database" (Lucey et al., 2010), which provides information on the peak intensity of each AU, and the "MMI Facial Expression Database" (Pantic et al., 2005), which details the onset, apex, and offset times of each Action Unit without including their intensity levels. After reviewing the available resources, we identified 21 potential datasets, which are accessible via the following GitHub link https://github.com/EvelynFan/AWESOME-FER?search=1#datasets. Finally, we decided to utilize the DISFA database for our study, due to it's fast response when requesting access to this database, once we got the access to the data we started validating that it was complete and the labels were also provided. Additionally, to enhance our understanding of facial behavior recognition using Action Units (AUs), we also revisited and replicated the methodologies described in the paper "Smile Detection in the Wild Based on Transfer Learning" (Guo et al., 2018). This study explores the efficacy of transfer learning for the task of smile detection across diverse, uncontrolled environments. The detailed analysis and results of this paper are available on the arXiv platform and can be accessed at https://arxiv.org/pdf/1802.02185.

Our research on this data reinforced the potential of transfer learning approaches in enhancing the accuracy of facial expression recognition, particularly in natural settings, which informed our decision to adopt similar methodologies in our project using the DISFA database. In this database, facial behavior is exhaustively annotated using anatomically based descriptions, and FACS action units.

The intensity of each action unit is annotated on a 6-point scale representing the intensity of the facial gesture, the distribution of these labels is shown in Figure 1.

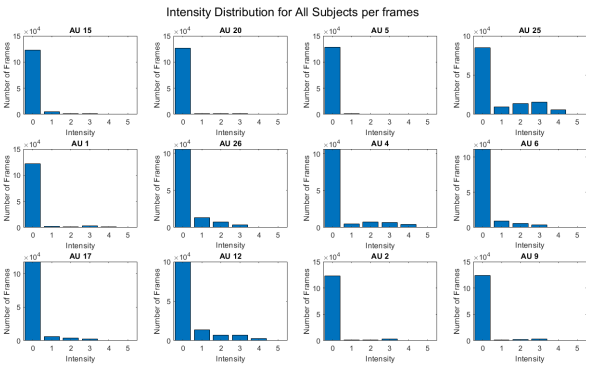

Figure 1: Distribution of the AU's intensities for all subjects, across all frames

This project aims to enhance baseline results by integrating three key features and employing innovative strategies, from data extraction to the development of robust classifiers for detecting action units and measuring their intensity levels. Additionally, we have implemented CNNs and RNNs to investigate the behavior of neural network approaches in this research domain. The pipeline we followed to achieve these objectives is depicted in Figure 2.

4 Objective

In this project, we will develop and assess a robust system for detecting facial behavior, utilizing the Denver Intensity of Spontaneous Facial Action (DISFA) dataset. Our approach includes detecting and quantifying facial actions via Action Units (AUs), employing foundational machine learning

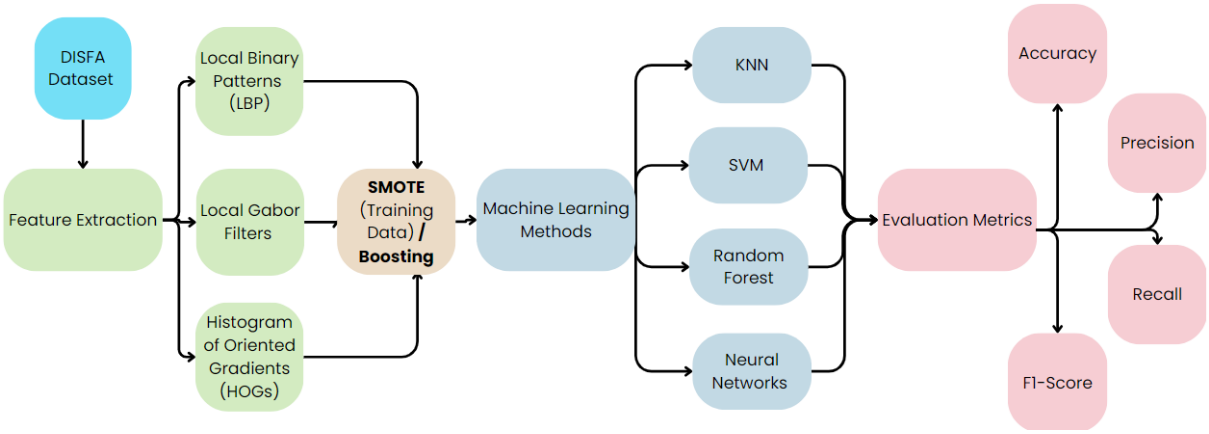

Figure 2: Project pipeline including data extraction, feature integration, classifier design and development, and neural network implementations.

techniques such as SVM, RF, and KNN, alongside more advanced methods like CNN and RNN. This will be achieved by extracting three image-texture features from videos and using labels that denote the intensity levels of the ratings. (Mavadati et al., 2013)

5 Methods

5.1 Feature Extraction

Before the development of a machine learning model using the provided dataset, it was observed that the dataset was imbalanced, with the majority of instances assigned a label of 0. In response to this challenge, several techniques were evaluated to address the imbalance. Notably, the Synthetic Minority Oversampling Technique (SMOTE) was identified as an effective solution. This technique operates by selecting examples that are close in the feature space, drawing a line between these examples, and generating a new sample at a point along this line (Chawla et al., 2002). SMOTE was specifically applied to the Local Binary Pattern Histogram (LBPH) feature, resulting in promising initial results, considering the now balanced nature of the database. In subsequent parts of this report, it is important to note that SMOTE was not applied to the Histogram of Oriented Gradients (HOG) and Gabor Filters features. Instead, for these two features the boosting technique was employed to improve classification results.

Before initiating the feature extraction process, it was essential to understand the positioning and distribution of landmarks across each subject over time. This comprehensive understanding is critical for the processing steps that are mentioned in the next steps of the pipeline. Figure 3 represents the dynamic changes and variability of landmark positions, these landmarks were used in the next steps for the feature extraction and subsequent analysis of facial features.

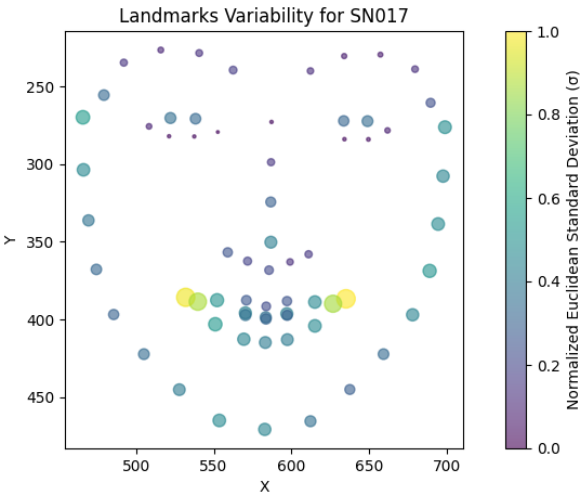

Figure 3: Distribution of facial landmarks over time during the stimulus for some of the subjects in the DISFA database.

Before performing the feature computation, it was necessary to perform video analysis and frame extraction on the data. These preliminary steps are detailed in the attached README file #1 in section 9.

5.1.1 Local Binary Pattern Histogram (LBPH)

In this project, the LBPH will serve as a key feature encoding the texture of the image, functioning as a descriptor that proficiently captures variations

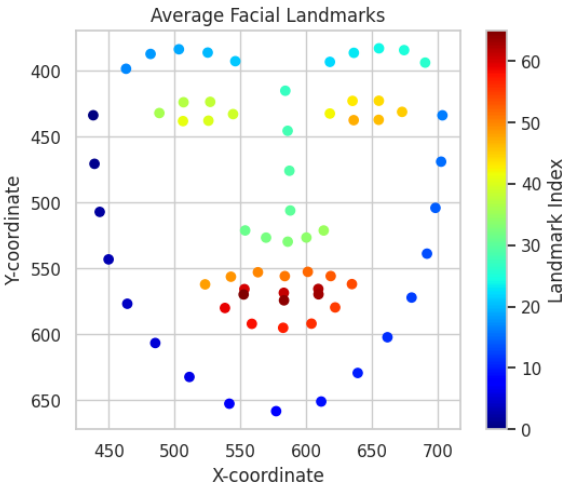

Figure 4: Average Facial Landmarks across all subjects

in the texture of the surface present in the facial expressions. This characteristic enables the differentiation of subtle changes in facial muscle movements.

LBPH has been extensively employed in applications related to biometrics and facial expression recognition (Zhao and Pietikäinen, 2007). In LBPH, for each pixel in the image, a central pixel is selected and its intensity is compared with that of its neighboring pixels, located within a specific radius. The comparison is that if the intensity of a neighboring pixel is greater than the intensity of the central pixel, a '1' is assigned, if it is less, a '0' is assigned. This bit sequence, generated around each center pixel, is converted to a decimal number representing a unique LBP value. In other words, the center pixel effectively functions as a threshold, assuming a zero threshold value, thus facilitating the classification and analysis of texture patterns in the image.

The steps followed for the computation of the Local Binary Pattern Histogram (LBPH) feature after extracting the landmarks and having the frames of the video were as follows: To calculate the LBPH features it was defined an area of 18x16 pixels around each of the landmarks specified for each of the frames. The data is extracted with parameters of a radius of 1 and 8 neighbors, which information is later stored in 59 bins. The resulting vector that contains this feature will be a vector of 4012, being 59 bins per each of the 68 landmarks.

Figure 5 shows the distribution of the (LBP) feature, where each bin represents a range of LBP values, contributing to the understanding of how frequently each texture range appears in the data. A higher bar indicates that a particular texture (or LBP value) is common in the images being analyzed. In this graph, each bin has a frequency, and as this figure averages all the LBP patterns from all landmarks and subjects it helps to represent the percentage of values that fall within the specific range of each bin. This suggests that the bin with the highest frequency in the graph represents a texture pattern that is commonly observed in the data.

- Workstation features where LBPH was computed
  - CPU: Ryzen 9 5900HS
  - RAM: 16GB DDR4
  - GPU: NVIDIA RTX 3070 8GB

5.1.2 Localized Gabor Filters

Gabor filters are effective in image processing due to their joint spatial and frequency domain representation. The Gabor filter is a sinusoidal plane wave (characterized by a certain frequency and orientation) module by a Gaussian envelope. This allows Gabor filters to capture information about object edges and texture changes within an image, which are crucial elements in tasks such as facial recognition.

A Gabor filter can be mathematically described as:

$$G(x, y : \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cdot \cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

where,

$x' = x\cos\theta + y\sin\theta$

$y' = -x\sin\theta + y\cos\theta$

$\lambda \rightarrow$ wavelength of sinusoidal factor

$\theta \rightarrow$ orientation of the normal to the parallel stripes of a Gabor function

$\psi \rightarrow$ phase effect

$\sigma \rightarrow$ standard deviation of Gaussian envelope

$\gamma \rightarrow$ Spatial aspect ration

The Gabor filter can simulate the visual perception of the human eye, particularly for orientation and scale. These are capable of highlighting facial

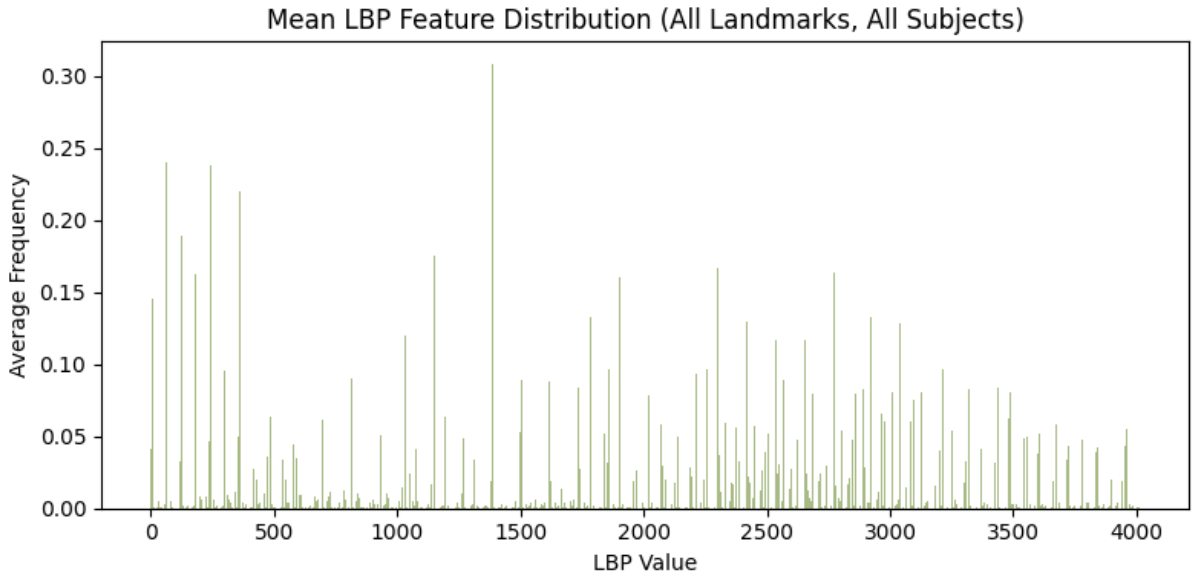

Figure 5: Distribution of Local Binary Pattern values across all landmarks and subjects.

features that are critical for recognizing expressions, identifying facial landmarks, or classifying facial attributes by capturing unique textural and edge information across different orientations and scales.(Mehrotra et al., 1992)

The figure 6 represents the distribution of Gabor filter responses across the subjects, illustrating the prominence and frequency of certain features detected by the filters. Each bin in the histogram corresponds to a specific range of values for the filter response.

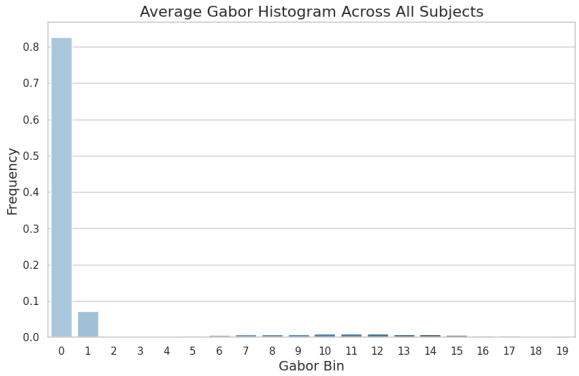

Figure 6: Distribution of Gabor Filters Responses Across Subjects

The histogram shows a significant concentration of responses in the lower bins, indicating that the majority of the Gabor filter outputs are low values. Further, the sparse values in higher bins suggest that high-frequency features, such as sharp edges or detailed textures are less common across the analyzed images.

5.1.3 Histogram of Oriented Gradients (HOGs)

Histogram of Oriented Gradients (HOGs) is a feature descriptor used for object detection. The technique counts the occurrences of gradient orientations in localized portions of an image.

The work principle involves:

- Calculating the gradient values for each pixel in the image, which provides the direction and strength of edges.
- The image is then divided into small connected regions known as cells.
- For each cell, a histogram of gradient directions is compiled. Each bin in the histogram represents an angle, and each gradient pixel in the cell votes for an orientation based on its intensity.
- To improve the accuracy, the histograms are normalized over larger, overlapping blocks of cells, which helps in compensating for changes in illumination or shadowing. (Dalal and Triggs, 2005)

Figure 7, shows the distribution of gradient orientations. The x-axis represents the orientations in degrees, the orientation angles of the gradient, ranging from 0 to 180 degrees. The histogram is divided into 8 bins, each bin representing an orientation range. The y-axis represents the average

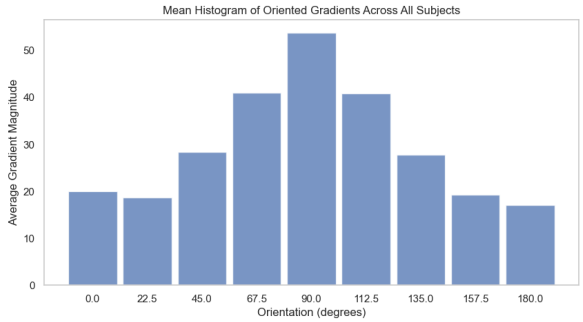

Figure 7: Mean Histogram of Oriented Gradients across all subjects

magnitude of the gradients for each orientation bin across all subjects. This measures the strength of edges corresponding to each orientation.

The figure shows certain orientations have higher average magnitudes, particularly around 67 degrees and 112 degrees. The 67.5-degree bin shows a prevalence of edges with this orientation across all subjects. This orientation represents features that are tilted, such as the slant of eyebrows, the natural contouring around the eyes and cheeks, and the tilt of the mouth in expressions, which is crucial for detecting facial expressions. The 112.5-degree orientation bin corresponds to diagonal features that include parts of the forehead, nose, and cheekbones.

6 Results

We separated the results according to the features.

6.0.1 Local Binary Pattern Histogram (LBPH)

For the model developed using Local Binary Pattern (LBP) features, it was initially necessary to apply undersampling and utilize the Synthetic Minority Oversampling Technique (SMOTE). These measures were taken to enhance the model's performance by improving the representation and influence of the minority classes within the dataset.

- Workstation features where the ML model for LBPH was computed
  - CPU: AMD Ryzen Threadripper PRO 5975WX 32-Cores
  - RAM: 256GB
  - GPU: NVIDIA RTX 6000

Implementation

- Data Preparation The training data was processed using the SMOTE technique to increase the examples of minority classes before proceeding to the testing phase.
  - Model Training
    The model employs a 5-fold cross-validation. For these ML models created with the LBPH feature, 80% of the data was used for training and 20% for testing.

Figure 8 compares the performance of the models created with this feature.

The models selected for this analysis were chosen strategically: Support Vector Machines (SVM) were employed as the baseline study does, and then we added Random Forest and K-Nearest Neighbors (KNN) models which have been widely utilized in the field of biometric signal analysis. (Mavadati et al., 2013) (Blanco-Ríos et al., 2024)

A multi-class classifier capable of distinguishing among 12 categories (representing 12 Action Units) and 6 labels (intensities ranging from 0 to 5) was created, and a 5-fold cross-validation was applied to these three models to ensure robustness and generalizability of the results.

Evaluation and analysis

The accuracy, precision, recall, and F1 score for each model for LBPH feature were computed and presented in Figure 8. These metrics were computed for each label in these multi-class classifiers and then averaged, providing a holistic view of each model's performance.

- **Accuracy:** All models show modest accuracy, with Random Forest slightly outperforming the others.
- **Precision:** The three main models show similar precision; however, RF and KNN outperform SVM in precision, meaning that these two models are more effective in correctly predicting positive instances among all instances they have classified as positive.
- **Recall:** SVM and RF perform better in terms of recall compared to KNN, suggesting that they capture a large proportion of the existing positive cases, minimizing the number of false negatives.
- **F1 Score:** Regarding this harmonic mean between precision and recall, RF shows a higher F1-Score performance, indicating that this

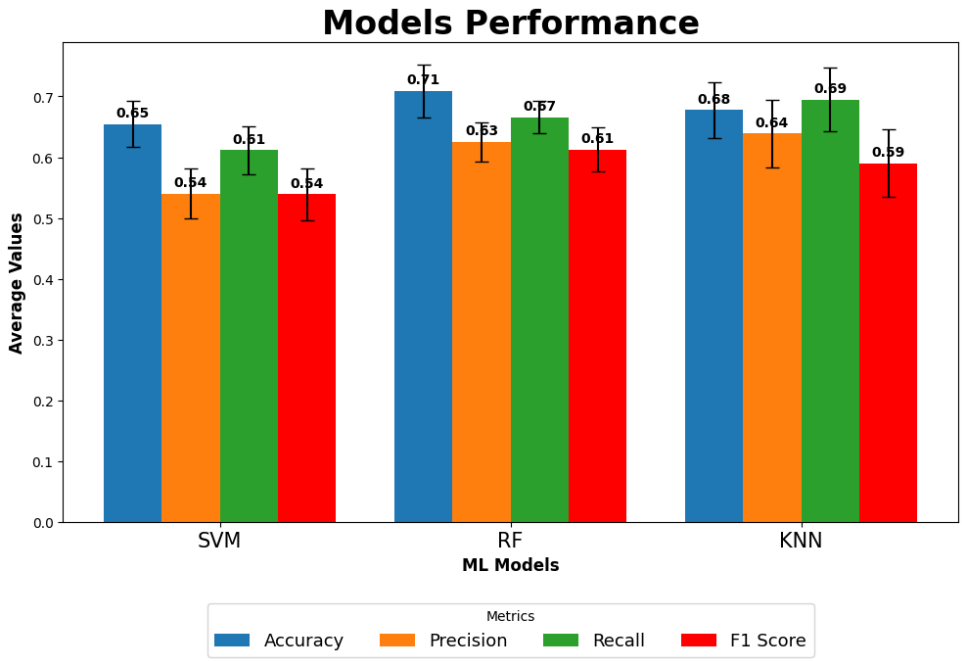

Figure 8: Performance of the ML models: Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN), verified through 5-fold cross-validation.

model presents a bit more balance than the others and that it is reliable both in precision and in identification capacity.

As a complement to Figure8, Table 1 presents the breakdown of the performance metrics for each of the 12 Action Units for the models a)SVM, b)RF, and c)KNN.

After reviewing these results, Random Forest (RF) emerges as the most effective among the three models, as it outperforms individual metrics and demonstrates strong and balanced performance across all metrics. Therefore, it proves advantageous in applications where both the precision of positive predictions and the comprehensive identification of true positive instances are crucial.

Once the basic ML models were analyzed, and the first results were obtained, we also wanted to try the performance the feature had when implementing a Recurrent Neural Network (RNN), the architecture for this Network is fully detailed in Section **6.0.2** . From the results, it was observed that the behavior of this RNN was not good. In Table 2 the average for the 12 categories (AUS) metrics of this RNN model for this feature is presented.

Based on the findings emerging from this subsection it can be noticed that the Recurrent Neural Network (RNN) exhibited suboptimal accuracy and therefore may not serve as a reliable model. On the contrary, the outcomes obtained through the application of basic machine learning classifiers are significantly superior, with Random Forest followed by K-Nearest Neighbor performing as the most effective models.

6.0.2 Localized Gabor Filter

We compared and evaluated multiple classifiers using the `MultiOuputClassifier` from the `sklearn.multioutput` module. The classifiers compared include Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (KNN), each adapted for multi-output classification tasks.

Implementation

- Data Preparation
  - The data was split into training and testing sets with `train_test_split`, using 30% of the data for testing to ensure a sufficient dataset size for both training and evaluation.
- Model Initialization
  - Multiple models are defined using `MultiOutputClassifier` to handle multi-label classification tasks, essential for scenarios where multiple outputs (like different AUs in facial recognition) are predicted simultaneously.

Table 1: Breakdown of the performance of different machine learning models for LBPH features by action unit

(a) Support Vector Machine (SVM)

AUs	Acc	Precision	Recall	F1 Score
AU1	0.69	0.58	0.62	0.59
AU2	0.67	0.55	0.59	0.54
AU4	0.64	0.55	0.60	0.55
AU5	0.68	0.51	0.61	0.51
AU6	0.62	0.52	0.57	0.50
AU9	0.74	0.61	0.65	0.61
AU12	0.59	0.51	0.56	0.55
AU15	0.64	0.52	0.60	0.52
AU17	0.64	0.52	0.60	0.52
AU20	0.64	0.51	0.53	0.53
AU25	0.69	0.58	0.66	0.59
AU26	0.63	0.46	0.56	0.46
Avg AU	0.65	0.54	0.61	0.54

(b) Random Forest (RF)

AUs	Acc	Precision	Recall	F1 Score
AU1	0.77	0.66	0.69	0.67
AU2	0.77	0.60	0.65	0.60
AU4	0.74	0.66	0.68	0.66
AU5	0.70	0.63	0.67	0.61
AU6	0.70	0.60	0.64	0.58
AU9	0.76	0.65	0.67	0.64
AU12	0.70	0.61	0.66	0.60
AU15	0.69	0.61	0.68	0.59
AU17	0.68	0.63	0.66	0.59
AU20	0.62	0.66	0.68	0.64
AU25	0.70	0.64	0.71	0.64
AU26	0.66	0.55	0.60	0.54
Avg AU	0.71	0.63	0.67	0.61

(c) K-Nearest Neighbor (KNN)

AUs	Acc	Precision	Recall	F1 Score
AU1	0.70	0.66	0.67	0.61
AU2	0.61	0.57	0.61	0.51
AU4	0.67	0.66	0.67	0.59
AU5	0.61	0.51	0.63	0.48
AU6	0.71	0.64	0.68	0.60
AU9	0.66	0.66	0.69	0.58
AU12	0.67	0.66	0.68	0.61
AU15	0.63	0.68	0.74	0.60
AU17	0.76	0.72	0.80	0.70
AU20	0.74	0.63	0.77	0.60
AU25	0.68	0.70	0.73	0.65
AU26	0.68	0.59	0.67	0.56
Avg AU	0.68	0.64	0.69	0.59

Table 2: RNN model performance on Action Unit (AU) classification with LBP feature displaying average accuracy, precision, recall, and F1-Score across AUs.

AUs	Acc.	Precision	Recall	F1 Score
Avg AU	0.21	0.1	0.17	0.09

- Model Training
  - Each model is trained on the training set, and performance is evaluated on the test set.

This feature was computed on a high-performance computing cluster, specifically using nodes equipped with dual Nvidia Tesla K40M GPUs.

Evaluation and analysis

- Four key metrics: accuracy, precision, recall, and F1 score for each model is calculated. These metrics are computed for each label (AU) in a multi-label setup and then averaged, providing a holistic view of each model's performance. The bar chart in Figure 9 displays the average performance metrics for each classifier used.
  - **Accuracy:** All models demonstrate high accuracy, with scores close to or above 80%. This shows that all models are generally effective at correctly identifying the true labels, both positive and negative, for various AUs.
  - **Precision:** Random Forest shows the highest precision, indicating that when it predicts an AU as present, it is very likely to be correct. The rest of the models show lower precision.
  - **Recall:** Recall values are notably lower than accuracy, suggesting that while the models are good at identifying negatives, they might be missing a significant number of true positive cases.
  - **F1 Score:** These are moderate, with Random Forest and Decision Tree performing slightly better than others.

The table 3 exhibits high performance but notably low recall and F1 score across most AUs. This indicates that while the model's prediction are reliable when it predicts an AU as present (high precision), it frequently misses detecting AUs that are actually present (low recall). AU5 and AU20 show exceptionally high precision and decent recall compared to other AUs, suggesting some AUs are better detected than other. The average performance shows balanced averaged accuracy and precision but struggle with recall and F1, which shows an overall conservative prediction behavior but likely leads to many false negatives.

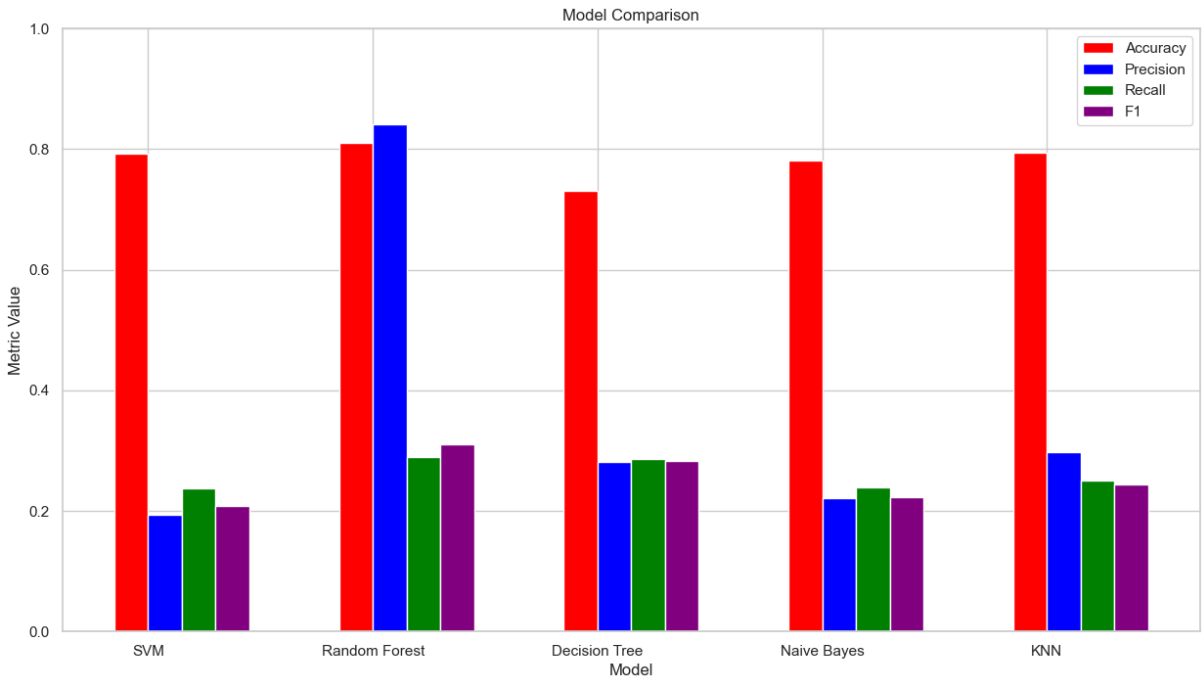

Figure 9: Model Comparisons of Gabor Features

Table 3: Performance Metrics for Each Action Unit (AU) using RandomForestClassifier

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.81	0.93	0.20	0.22
AU2	0.95	0.98	0.21	0.24
AU4	0.76	0.90	0.21	0.22
AU5	0.99	0.99	0.37	0.40
AU6	0.88	0.94	0.29	0.31
AU9	0.86	0.95	0.21	0.23
AU12	0.84	0.86	0.28	0.30
AU17	0.99	0.66	0.35	0.36
AU20	0.99	0.99	0.38	0.43
AU25	0.42	0.57	0.28	0.29
AU26	0.38	0.37	0.35	0.35
Avg AU	0.81	0.83	0.28	0.31

Table 4: Performance Metrics for Each Action Unit (AU) using MultiOutput AdaBoost Classifier

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.9529	0.9191	0.7798	0.8397
AU2	0.9857	0.9493	0.7183	0.8064
AU4	0.9781	0.9364	0.8248	0.8646
AU5	0.9944	0.6648	0.5057	0.5597
AU6	0.9582	0.9353	0.8780	0.9047
AU9	0.9881	0.9797	0.6962	0.7772
AU12	0.9349	0.9173	0.8680	0.8907
AU17	0.9960	0.9711	0.6887	0.7709
AU25	0.9267	0.9299	0.8389	0.8776
AU26	0.9436	0.9474	0.9470	0.9472
Avg AU	0.96	0.91	0.77	0.82

Table 4 demonstrates significantly better recall and F1 scores compared to table 3, indicating a stronger ability to detect AUs when they are present. AUs like AU4, AU6, and AU12 show particularly strong performance across all metrics, with no AU showing drastically poor metrics. The average performance excels in all metrics showing its robustness and effectiveness in detecting a broad range of AUs accurately and consistently.

The table 5 shows very mixed results with some AUs like AU5, and AU17 exhibiting high accuracy and F1 scores, while others like AU1, and AU25 perform poorly across all metrics. The overall lower average scores across metrics show challenges in model training. The RNN has 5 layers in total and 4 are hidden ones.

- 2 LSTM
- 2 Dropout
- 1 Dense

Further, we introduced Optuna as a hyperparameter optimization technique. This optimiza-

Table 5: Performance Metrics for Each Action Unit (AU) using RNN

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.01	0.002	0.16	0.005
AU2	0.94	0.15	0.16	0.16
AU4	0.90	0.15	0.16	0.15
AU5	0.99	0.33	0.33	0.33
AU6	0.80	0.20	0.25	0.22
AU9	0.95	0.49	0.16	0.16
AU12	0.08	0.02	0.25	0.03
AU17	0.99	0.49	0.50	0.49
AU25	0.03	0.007	0.20	0.01
AU26	0.44	0.14	0.33	0.20
Avg AU	0.61	0.16	0.25	0.17

tion provides the best parameter set and the visualization of the result. The best parameters found was: `Best trial:'lstm_units': 150, 'dropout_rate': 0.1608366146230273, 'learning_rate': 0.0007330124021490447`.

The MultiOutput AdaBoost Classifier outperforms the two methods significantly, offering robust options for AU detection with high reliability and effectiveness.

6.0.3 Histogram of Oriented Gradients (HOGs)

Here, we used the same approach that was used in Gabor Filters to compare and evaluate multiple classifiers. The same implementation technique was used for data preparation, initialization, and training. Also, the same cluster was used to compute this feature as well.

Evaluation and Analysis

The four metrics: accuracy, precision, recall, and F1 score for each model are calculated. Similarly, these metrics are computed for each label in a multi-label setup and then averaged. The bar chart in Figure 10 displays the average performance metrics for each classifier used.

- **Accuracy:** Most of the model demonstrated high accuracy of more than 90%, except for Naive Bayes. KNN performed the best among all. This shows that most of the models are generally effective at correctly identifying the true labels, both positive and negative, for various AUs.
- **Precision:** KNN shows the highest precision as well, indicating it predicts an AU as present, it is very likely to be correct.
- **Recall:** Recall values of SVM and KNN are high which means it returns most of the relevant results.
- **F1 Score:** SVM and KNN's F1 scores are high in comparison to other models which indicates a well-balanced performance of the two models.

The table 6 outlines the performance of a Random Forest Classifier when applied to AU detection using HOGs. Accuracy is high across AUs, which shows that the model is effective in correctly classifying the presence or absence of AUs. Precision is high for AU1, AU6, and AU12 which implies that when the model predicts these AUs as present, is very likely to be correct. The recall value is not notably varied, with lower values like AU2 and AU17, indicating some AUs are often missed by the model. High F1 scores for AUs like AU26 shows well-balanced precision and recall, while lower scores for AU2 and AU20 show imbalance.

Table 6: Performance Metrics for Each Action Unit (AU) using Random Forest Classifier

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.88	0.90	0.45	0.53
AU2	0.97	0.65	0.30	0.35
AU4	0.88	0.88	0.48	0.57
AU5	0.99	0.65	0.38	0.42
AU6	0.92	0.92	0.54	0.64
AU9	0.93	0.92	0.51	0.62
AU12	0.92	0.96	0.62	0.69
AU17	0.99	0.66	0.36	0.39
AU20	0.99	0.33	0.33	0.33
AU25	0.88	0.92	0.63	0.66
AU26	0.93	0.79	0.91	0.92
Avg AU	0.96	0.91	0.50	0.56

In table 7, shows the evaluation of Histogram of Oriented Gradients using K-NN Classifier, where we found the highest results. The classifier achieves high accuracy across all AUs, with an average accuracy of 0.97. This suggests that the classifier is highly effective in correctly identifying the presence or absence of AUs. The precision and recall values are also notably high, indicating that the classifier not only accurately identifies the relevant AUs

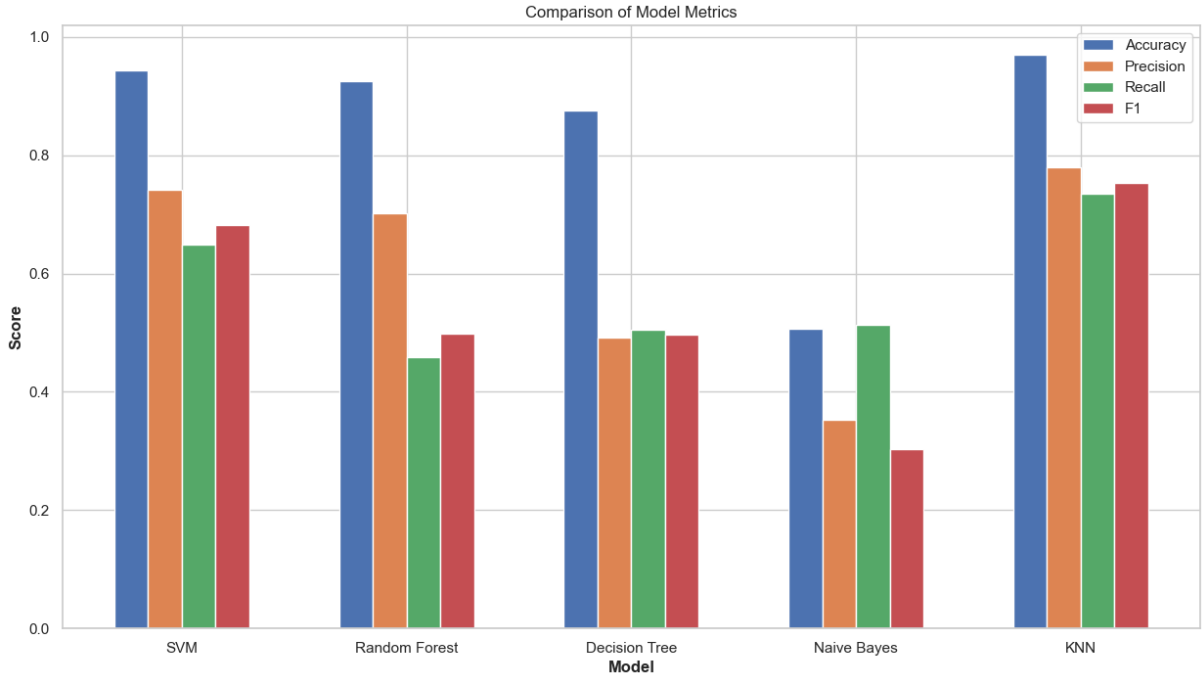

Figure 10: Model Comparisons of HOGs

Table 7: Performance Metrics for Each Action Unit (AU) using K-NN Classifier

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.95	0.82	0.78	0.80
AU2	0.98	0.76	0.68	0.72
AU4	0.95	0.84	0.80	0.82
AU5	0.99	0.94	0.72	0.72
AU6	0.97	0.90	0.88	0.88
AU9	0.97	0.85	0.83	0.83
AU12	0.97	0.90	0.89	0.89
AU17	0.99	0.64	0.62	0.62
AU20	0.99	0.71	0.62	0.62
AU25	0.94	0.90	0.88	0.88
AU26	0.96	0.96	0.96	0.96
Avg AU	0.97	0.84	0.77	0.79

but also minimizes false positives and false negatives effectively. AU26 stands out with the highest precision, recall, and F1 score of 0.96, suggesting that this AU's characteristics are particularly well-captured by the feature.

We couldn't run SMOTE, Neural Networks on HOG as the computation time was too high to calculate it. Further, the results were so high using K-NN, running other classifers were not feasible.

7 Innovative Contributions

As an alternative approach for analyzing the dataset and constructing machine learning models, we have combined the features of Localized Gabor Filters and Histograms of Oriented Gradients (HOGs).

In Figure 11, it is shown the t-SNE for the combined two features that presented the best performance, this combination is being displayed in two-dimensional space. The visualization shows several

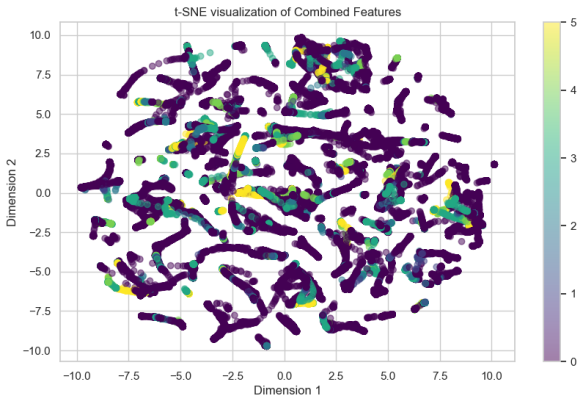

Figure 11: t-SNE Visualization of Localized Gabor Filter and HOGs where the color bar, shows the intensity of the AUs

distinct clusters scattered across the plot. These clusters are indicative of groups of data points that are similar to each other in terms of their underlying features. The data points are not uniformly

distributed but form several dense regions with varying degrees of separation. This separation can suggest that the data owns intrinsic groupings that correspond to different categories, there are also isolated points and small groupings of points away from the main clusters. These could be unique instances.

Results from the models created with this data are presented in figure 12, which compares the performance of two machine learning models, Random Forest and K-Nearest Neighbors (KNN), across the four key metrics: Accuracy, Precision, Recall, and F1 Score.

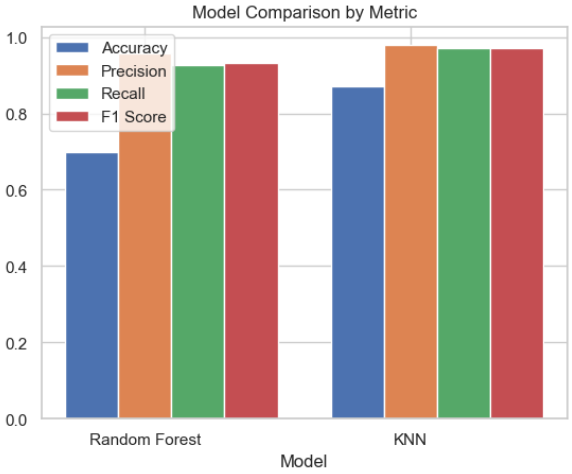

Figure 12: Combined Features model comparison

Both models show high accuracy, but KNN appears to have slightly higher accuracy than the Random Forest model. This suggests that KNN may be better at correctly identifying both positive and negative classes across all labels. Precision is also high for both models, with KNN showing a marginal advantage. This indicates that when either model predicts an AU as present, it is likely to be correct. Recall measures the ability of the models to find all the relevant instances of each class. Both models perform comparably on this metric, shows that they are equally capable of identifying most of the positive samples in the dataset. The F1 Score is high for both models, indicating a balanced performance between precision and recall.

The performance metrics for each Action Unit (AU) using Random Forest (RF) and K-Nearest Neighbors (KNN) are presented in Table 8 and Table 9, respectively. It is important to note that this specific analysis was limited to only five AUs due to the extensive computational time required relative to the time available.

Table 8: Performance Metrics for Each Action Unit (AU) using RF (combined features)

AU	Acc.	Precision	Recall	F1 Score
AU1	0.896	0.928	0.753	0.832
AU2	0.897	0.880	0.942	0.910
AU3	0.934	0.911	0.968	0.939
AU4	0.917	0.972	0.359	0.525
AU5	0.968	0.974	0.292	0.449
Avg AU	**0.935**	**0.944**	**0.719**	**0.776**

Table 9: Performance Metrics for Each Action Unit (AU) using KNN (combined features)

AU	Acc.	Precision	Recall	F1 Score
AU1	0.948	0.935	0.910	0.922
AU2	0.949	0.955	0.952	0.953
AU3	0.979	0.979	0.980	0.979
AU4	0.975	0.921	0.880	0.900
AU5	0.990	0.911	0.865	0.887
Avg AU	**0.973**	**0.950**	**0.931**	**0.940**

7.0.1 Feature Extraction & ML Approach

As detailed in the report, the approach for feature extraction diverged significantly from standard methods as we implemented a system involving 68 facial landmarks, enhanced by two additional markers to compute the features. This process was executed using Python, and supplemented by functions from the OpenCV library.

For the application of machine learning techniques, first, we read different approaches that were used for projects focused on biometric data (Blanco-Ríos et al., 2024) (Mavadati et al., 2013) (Dino and Abdulrazzaq, 2019)(Zhao and Pietikäinen, 2007), and implemented some of them. These were chosen strategically, and it's important to mention that the baseline model does not incorporate more than one ML model.

A comparative analysis with the baseline model indicated that the Random Forest model demonstrated substantial efficacy in classifying the dataset. Furthermore, we conducted an analysis combining features to evaluate the behavior of the models under these conditions. The four key metrics used to analyze model performance yielded exceptionally high values, underscoring the robustness of our approach.

7.0.2 Convolutional Neural Network

Following the initial approach with basic machine learning models and RNNs, we proceeded

to explore the application of Convolutional Neural Networks (CNNs) on the raw images extracted from the video footage. This analysis is focused on the regions of the face defined by the landmarks. The architecture used for the creation of this Neural Network is the following: This model contains two convolutional layers created with `"nn.Conv2d()"`, both utilize the ReLU activation function implemented with `"nn.ReLU()"`. Subsequently, both layers employ Dropout as the regularization method to prevent overfitting `"nn.Dropout(0.25)"`, this method prevents overfitting by randomly deactivating a fraction of its inputs. Finally, the `"nn.MaxPool2d()"` function is applied to perform pooling on the data and reduce dimensions. Then, `"nn.Flatten()"` is applied to convert the vector to one dimension, and when `"loss.backward()"` is executed, gradients of the loss are calculated and are stored in `".grad"`. Then, `"optimizer.step()"` uses these gradients to update the weights of the model using the optimization algorithm called Adam `"optim.Adam()"`. As part of this neural network, the `Optuna library` is employed to perform hyperparameter tuning. Optuna tunes the following two parameters.

- **Batch Size**: Optuna suggests and finds the best batch size for data during training and validation. This tuning can influence the stability and speed of training.

- **Learning Rate (lr)**: For this parameter, it suggests a learning rate for the Adam optimizer. The learning rate is a floating-point value that varies between 1×10^{-5} and 1×10^{-1} on a logarithmic scale. This variable addresses how the model converges to a solution by determining the speed at which the model updates the weights in response to the calculated error.

After tuning the parameters using Optuna, confusion matrices, and performance metrics were generated for the Convolutional Neural Network (CNN) to evaluate its effectiveness. Two of the confusion matrices generated for this neural network are illustrated in Figure 13, and in figure 14 the average performance of the CNN model across the 12 AUs is presented.

Table 10 presents the performance of the model for each Action Unit (AU), including the four main metrics used throughout this study to analyze model performance.

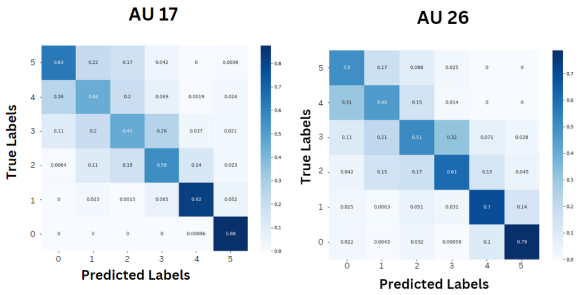

Figure 13: Confusion matrices illustrating the performance of the CNN model on two Action Units, exemplifying a subsample of the material analyzed.

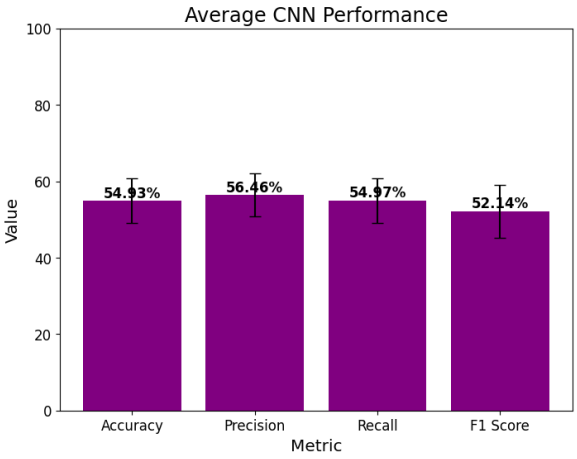

Figure 14: Performance of the two-layer Convolutional Neural Network, using raw images as input.

Table 10: Performance of the CNN model on Action Unit (AU) classification using raw images from video frames as input, presenting accuracy, precision, recall, and F1-Score metrics across AUs.

AUs	Acc.	Precision	Recall	F1 Score
AU1	0.49	0.50	0.49	0.45
AU2	0.57	0.56	0.57	0.54
AU4	0.47	0.50	0.48	0.42
AU5	0.54	0.55	0.54	0.50
AU6	0.58	0.59	0.59	0.55
AU9	0.52	0.55	0.52	0.50
AU12	0.51	0.55	0.51	0.48
AU15	0.66	0.70	0.66	0.65
AU17	0.64	0.63	0.64	0.63
AU20	0.54	0.55	0.54	0.51
AU25	0.47	0.50	0.47	0.45
AU26	0.60	0.59	0.60	0.58
Avg AU	0.55	0.56	0.55	0.52

Regarding the CNN's performance, the model appears to be moderately successful, with average

metrics $\approx 50\%$ range. The best-performing AUs (AU15 and AU17) show that the model can achieve relatively high accuracy and precision in certain cases. However, the overall performance suggests there might be room for improvement, which could be in the model architecture or parameter tunning that can help the model become more robust and reliable for practical applications.

8 Conclusion

This study explored the efficacy of various machine learning techniques for recognizing spontaneous facial expressions using the DISFA dataset. Initially, we examined traditional and well-known models, including Support Vector Machine (SVM), Random Forest (RF), and K-Nearest Neighbors (KNN). Subsequently, we integrated neural network models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) into our analysis.

The incorporation of RNNs with texture-based features did not improve the performance metrics as anticipated, indicating that the machine learning models we implemented performed better in classification tasks. For these classification tasks, particularly for the Local Binary Pattern (LBP) feature, it was necessary to employ the SMOTE technique to balance features to enhance the model's performance.

Another insight from our project involved diving into the area of feature combination, where two of the most accurate features, Histogram of Oriented Gradients (HOG) and Gabor Filters, were combined, and t-SNE was applied to them. The results demonstrated that this model achieved a really good performance across the four key metrics as it is shown in Table 9.

To enhance the basic ML models that utilize just one feature, we propose focusing on the most representative landmarks to reduce the dimensionality of feature vectors and concentrate on the most relevant features in future work. Additionally, working with a more balanced dataset would be beneficial, especially since it became apparent for the LBP feature that the infrequent and low appearance of labels with the value of 5 hinder the model's ability to make accurate predictions, this is the reason why the SMOTE technique was applied.

In conclusion, this research integrates basic ML models with advanced neural networks, laying groundwork for future studies in this area. Throughout the development of this project, we explored the integration of additional modalities and the practical application of these models, on the other hand, our research opens new avenues for understanding and leveraging spontaneous facial expressions across various domains, offering promising directions for further advancements in the machine learning and deep learning approaches for spontaneous facial gesture recognition.

9 Codes

The ReadMe File #1 and the codes that are explained in this file are contained in the following link: https://drive.google.com/drive/folders/1jeBfg3dppakX5WCOc7MIhaOEzVKN7fNx?usp=sharing

The second readme file with the codes can be accessed via the following link: https://www.icloud.com/iclouddrive/0155K-zP_ajlj0qa__CUJx-6g#ML_Project.

References

What is the k-nearest neighbors algorithm? | IBM — ibm.com. https://www.ibm.com/topics/knn#:~:text=the%20next%20step-,What%20is%20the%20KNN%20algorithm%3F,of%20an%20individual%20data%20point.

Nawaf Hazim Barnouti et al. 2016. Improve face recognition rate using different image pre-processing techniques. *American Journal of Engineering Research (AJER)*, 5(4):46–53.

Miguel Alejandro Blanco-Ríos, Milton Osiel Candela-Leal, Cecilia Orozco-Romo, Paulina Remis-Serna, Carol Stefany Vélez-Saboyá, Jorge de Jesús Lozoya-Santos, Manuel Cebral-Loureda, and Mauricio Adolfo Ramírez-Moreno. 2024. Real-time eeg-based emotion recognition for neurohumanities: perspectives from principal component analysis and tree-based algorithms. *Frontiers in Human Neuroscience*, 18.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Jeffrey F Cohn, Zara Ambadar, and Paul Ekman. 2007. Observer-based measurement of facial expression with the facial action coding system. *The handbook of emotion elicitation and assessment*, 1(3):203–221.

N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and*

Pattern Recognition (CVPR'05), volume 1, pages 886–893 vol. 1.

Hivi Ismat Dino and Maiwan Bahjat Abdulrazzaq. 2019. Facial expression classification based on svm, knn and mlp classifiers. In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, pages 70–75. IEEE.

Paul Ekman and Wallace V Friesen. 1978. *Facial action coding systems*. Consulting Psychologists Press.

Xin Guo, Luisa F. Polanía, and Kenneth E. Barner. 2018. Smile detection in the wild based on transfer learning.

Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. 2014. Facial expression recognition via a boosted deep belief network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1805–1812.

Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. 2010. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE.

S. Mohammad Mavadati, Mohammad H. Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F. Cohn. 2013. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160.

R. Mehrotra, K.R. Namuduri, and N. Ranganathan. 1992. Gabor filter-based edge detection. *Pattern Recognition*, 25(12):1479–1494.

Maja Pantic, Michel Valstar, Ron Rademaker, and Ludo Maat. 2005. Web-based database for facial expression analysis. In *2005 IEEE international conference on multimedia and Expo*, pages 5–pp. IEEE.

Erika L Rosenberg and Paul Ekman. 2020. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press.

Diaa Salama AbdELminaam, Abdulrhman M Almansori, Mohamed Taha, and Elsayed Badr. 2020. A deep facial recognition system using computational intelligent algorithms. *Plos one*, 15(12):e0242269.

Lijun Yin, Xiaozhou Wei, Yi Sun, Jun Wang, and Matthew J Rosato. 2006. A 3d facial expression database for facial behavior research. In *7th international conference on automatic face and gesture recognition (FGR06)*, pages 211–216. IEEE.

Zhihong Zeng, Maja Pantic, Glenn I Roisman, and Thomas S Huang. 2007. A survey of affect recognition methods: audio, visual and spontaneous expressions. In *Proceedings of the 9th international conference on Multimodal interfaces*, pages 126–133.

Guoying Zhao and Matti Pietikäinen. 2007. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 29:915–28.

A Appendix

A.1 Machine Learning Techniques

We will compare several machine learning techniques in classifying facial behaviors.

- **Support Vector Machine (SVM):** SVM is a powerful and versatile supervised learning model used for classification and regression tasks. It works by finding a hyperplane that best separates the data into different classes. The goal is to maximize the margin between the data points of the different classes, using support vectors and kernel tricks to handle non-linear boundaries.
- **Random Forest (RF):** Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. RF improves predictive accuracy and controls over-fitting by averaging multiple trees that individually suffer from high variance.
- **K-Nearest Neighbors (KNN):** KNN algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.(ibm)
- **Neural Networks:** To learn the complex patterns and perform feature extractions, neural networks will be included in the comparison. We will be using Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN) in this project.

A.2 Optuna

Optuna is an open-source hyperparameter optimization framework designed to automate and improve the process of finding the most effective hyperparameters for machine learning models. It offers a straightforward and flexible way to define the search space for parameters and uses algorithms to explore this space efficiently.

- **Study:** Represents an optimization task, a single run of hyperparameter optimization using

Optuna. This might contain multiple trials, each of which tests a particular set of hyperparameters and records the resulting performance.

- **Trial:** Corresponds to a single evaluation of the objective function with a specific set of hyperparameters. Optuna internally suggests values for parameters based on the defined search space and the history of previous trials.
- **Samplers:** This decides which hyperparameter to test next. It uses various strategies to explore the parameter space, from simple random sampling to Bayesian optimization, Tree-structured Parzen Estimator (TPE), and CMA-ES algorithm.
- **Pruners:** This is used to stop the evaluation of a trial early if it is determined the trial is unlikely to result in a better outcome than an already achieved trial. This reduces computational waste and focuses on resources on more promising parameter combinations.

A.3 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for dimensionality reduction, particularly well-suited for the visualization of high-dimensional datasets. Commonly used in the analysis of complex data such as gene expression, financial data, or image data, t-SNE helps to visualize how data is grouped and to identify patterns that are difficult to detect in high-dimensional space.

Key Features

- **Non-linear Dimensionality Reduction:** Unlike PCA (Principal Component Analysis), which is linear, t-SNE captures complex non-linear structures in the data, which makes it more effective for datasets where the relationship between points involves non-linear interactions.
- **Probabilistic Approach:** t-SNE converts high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The similarity of datapoint x_j to datapoint x_i is the conditional probability $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i.

Working

- **Compute Similarities in High-dimensional Space:** For each pair of data points in the high-dimensional space, compute the probabilities that represent similarities. This is based on the Euclidean distance between the points, adjusted by a user-defined perplexity parameter that roughly determines the local neighborhood size of each point.
- **Map to a Lower-dimensional Space:** Initialize the points in a lower-dimensional space and compute similar pairwise probabilities using the t-distribution.
- **Optimize Positions:** The positions of the points in the lower-dimensional space are optimized iteratively through a gradient descent method to minimize the Kullback-Leibler divergence between the probability distributions in the high-dimensional and lower-dimensional spaces.