Face Emotion Detection

Group: Team 1 Machine Learning Final Project Report

Anuja Kolse,[1] Saideep Samineni,[1] Aishwarya Kumar,[1] Rohith Chandra Kandambeth,[1]
Khoury College of Computer Sciences [1]
Northeastern University
Boston, USA

Abstract

This study was conducted by Team 1 to address the challenge of recognizing human emotions from facial expressions using machine learning techniques. Given the increasing significance of human-computer interaction, an accurate and efficient model for emotion detection can greatly enhance communication, empathy, and user interface adaptability. We present a comprehensive approach employing the Support Vector Machine (SVM) algorithm tailored for high-dimensional facial data. Utilizing two prominent datasets, Expression in-the-wild (ExpW) and FER2013, our model classifies seven fundamental emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. We adopt cross-validation methods to optimize performance and mitigate overfitting, ensuring robustness and reliability of our emotion classification system. Evaluation metrics including accuracy, precision, recall, F1-score, and the Receiver Operating Characteristic (ROC) curve demonstrate the model's effectiveness. Our findings suggest significant potential for applications in various domains such as mental health, personalized customer service, and interactive entertainment.

Introduction

In today's digital era, the ability to discern human emotions from facial expressions using machine learning techniques is more than a technical endeavor; it is a requisite for enhancing interfaces in human-computer interactions (HCI). This capability is crucial across various domains including mental health diagnostics, personalized customer interactions, and adaptive learning environments. The complexity of human emotions, which can be subtly expressed and highly subjective, poses significant challenges for automated systems. These challenges are compounded by the variations in facial expressions caused by cultural, contextual, and individual differences. Our project addresses these issues by implementing and comparing different configurations of the Support Vector Machine (SVM) algorithm, renowned for its robust performance in classification tasks involving high-dimensional data.

The rationale behind selecting SVMs for this task lies in their effectiveness in handling nonlinear relationships and their ability to model complex boundaries between different emotional states. To systematically assess the capabilities of our models, we utilize two comprehensive datasets: Expression in-the-wild (ExpW) and FER2013. These datasets include a wide range of emotional expressions captured in diverse settings, making them ideal for training and testing our models. This comparative study is designed to explore various SVM configurations and their impact on the accuracy and reliability of emotion classification, providing insights into the strengths and limitations of each approach within the context of HCI.

Problem Statement

The task of detecting human emotions from facial expressions can be formally conceptualized as a multi-class classification problem. The primary objective is to develop a predictive model that can accurately identify the specific emotional state of an individual from a given facial image. This challenge can be articulated as follows:

Given a dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, where:

- $\mathbf{x}_i$ represents the feature vector for the i-th facial image, encompassing various characteristics extracted from the image such as facial landmarks, expression intensities, and other relevant facial attributes.
- y_i is the categorical target variable for the i-th facial image, with y_i belonging to one of the predefined classes representing distinct emotions (e.g., Angry, Happy, Sad, Surprise, Neutral, etc.).

The goal is to construct a predictive machine learning model $f : \mathbf{X} \rightarrow Y$, where f is a function that maps the space of input features $\mathbf{X}$ to the output space Y. Specifically, the model f seeks to predict the emotional state y_i based on the input facial features $\mathbf{x}_i$.

Evaluation criteria

We have selected the following metrics for assessing the performance of machine learning models in facial emotion detection.

- **Accuracy**: Evaluates the overall effectiveness of the model in making correct emotion classifications, calculated by the ratio of correct predictions to total predictions. However, due to possible class imbalance, accuracy might not fully reflect the model's performance.

- **Precision**: Measures the model's capability to correctly identify a specific emotion without incorrectly labeling other emotions as that one, highlighting the importance of minimizing false positives.
- **Recall**: Focuses on capturing as many true instances of a specific emotion as possible, essential in applications needing accurate recognition of certain emotions for timely responses.
- **F1 Score**: Provides a balance between Precision and Recall, useful in scenarios where both false positives and negatives are critically impactful.
- **ROC Curve and AUC-ROC Score**: Assesses the model's ability to differentiate between emotions correctly versus making false identifications, with a high AUC-ROC Score indicating strong discriminative power.

Workflow Overview

Our emotion detection model workflow comprises several structured phases: Data Collection, Data Cleaning, Feature Engineering, and Normalization. Feature representations are then extracted using FaceNet and DeepFace models, where facial embeddings are computed and stored. These embeddings serve as input for training machine learning models, including support vector machines (SVM) and multi-layer perceptrons (MLP), which classify the faces based on the learned representations.

Significant emphasis is placed on hyperparameter tuning, particularly for the SVM Model, to optimize parameters such as the kernel, C, and gamma values. Additionally, data augmentation techniques such as rotations, scaling, and flips have been employed to increase the robustness of the models by introducing variability into the training process.

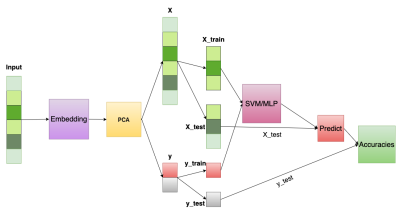

Figure 1: A diagrammatic representation of our model workflow.

The performance of these models are evaluated using accuracy metrics and confusion matrices to assess their effectiveness in correctly identifying or classifying faces across diverse datasets.

Data Collection

For this project, we have collected facial expression datasets, primarily focusing on two publicly available sources known for their robustness and wide usage in the research community:

- **The Expression in-the-Wild (ExpW+):** The Expression in-the-Wild (ExpW) Dataset is a and diverse collection of facial images curated to capture spontaneous and unscripted facial expressions exhibited by individuals in real-world scenarios. Dataset contains 91,793 faces manually labeled with expressions.
- **Facial Expression Recognition 2013 (FER2013):** Comprising approximately 30,000 grayscale images in a fixed resolution of 48x48 pixels, this dataset is categorized into the same seven emotional states as CK+.

These datasets have been preprocessed and transformed into a consistent format to ensure uniformity across both collections. This includes standardizing the labels to maintain consistency. The training set is used to fit the model, while the testing set is reserved to evaluate the model's performance.

Data Preparation

FER Dataset - The data is organized within an "archive" directory, which contained two subdirectories: "train" and "test". Each subdirectory is further divided into folders named according to the emotion class they represent. This organization facilitated the use of folder names as labels for the images, simplifying the process of associating each image with its corresponding emotion category.

Emotion Classes

The dataset labels the images under seven emotion categories as follows:

Angry, Disgusted, Fearful, Happy, Neutral, Sad, Surprise.

These categories are crucial for training the classification models, as they represent the emotional state depicted in each image.

Data Cleaning

During the data cleaning phase, Label Manipulation was performed on ExpW Dataset to maintain consistency across both the datasets. Labels from ExpW dataset are modified by swapping values (5, 6, 4) to new ones (6, 4, 5) to match the classes in FER dataset.

Feature Engineering

Feature engineering is a crucial step in the preparation of our dataset, involving several strategic actions to enhance model training and prediction accuracy:

- **Label Encoding:** We transformed categorical target labels into numerical format using Label Encoding. This process assigns a unique integer to each category, which is necessary because most machine learning models require numerical input for efficient training.
- **Scalar Scaling:** Scalar scaling, or feature scaling, is a crucial preprocessing step that involves adjusting the scale of features in a dataset to a common scale. We have utilized StandardScaler to normalize the training data. This ensures that each feature contributes equally to the analysis by giving them all a mean of zero and a standard deviation of one.

- **Data Storage:** Post feature extraction, the embeddings along with their respective labels are stored in a structured CSV format. This is to ensure that the data is organized and easily accessible for further processing or model training without having to redo the feature extraction.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical method that transforms a set of potentially correlated variables into a set of linearly uncorrelated variables called principal components.

Selection of Principal Components

Principal components are chosen based on their eigenvalues, with the largest corresponding to the directions capturing the most variance. The process involves selecting the eigenvectors with the highest eigenvalues to form the principal components.

Dimensionality Reduction Implementation

Dimensionality reduction through PCA involves retaining only those principal components that account for the most significant variance, effectively reducing the dataset to a lower-dimensional space while preserving essential information. This is typically implemented using libraries like Scikit-Learn.

Models and Methodologies

Feature representations are extracted using FaceNet and DeepFace models, where facial embeddings are computed and stored. These embeddings serve as input for training machine learning models, including support vector machines (SVM) and multi-layer perceptrons (MLP), which classify the faces based on the learned representations.

FaceNet

FaceNet is a facial recognition system that uses deep learning algorithms to map facial features into Euclidean space, where distances represent face similarity. Developed by Google researchers in 2015, it has achieved high results in many benchmark datasets, such as Labeled Faces in the Wild (LFW) and Youtube Face Database.

Architecture FaceNet is built upon a deep convolutional network architecture. Although the original implementation can utilize any deep learning architecture, the authors primarily employed the Inception model as the backbone for feature extraction. This choice was due to Inception's effectiveness in handling image data with minimal computational resources while still ensuring a high degree of accuracy.

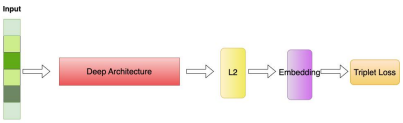

Figure 2: A diagrammatic representation of the FaceNet Architecture.

Embedding Space FaceNet directly trains its network to output a compact 128-dimensional embedding of a face image. These embeddings are designed so that distances between them (usually Euclidean) correspond to a measure of face similarity. Thus, faces of the same individual have lower distances between their embeddings compared to faces from different individuals.

Model Explanation The model employs either the ZF-Net or Inception Network as its foundational architecture. Additionally, it incorporates multiple 1x1 convolutional layers to reduce the number of parameters. These deep learning architectures generate an embedding, denoted as f(x), of the image, upon which L2 normalization is subsequently applied. These normalized embeddings are input into the loss function for loss computation.

The objective of this loss function is to minimize the squared Euclidean distance between embeddings of the same identity, regardless of variations in image conditions or poses, while maximizing the squared distance between embeddings of different identities. To achieve this, a specialized loss function known as Triplet loss is utilized.

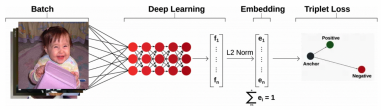

Figure 3: Working of FaceNet Architecture.

VGG Model

VGG (Visual Geometry Group) is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers. The architecture was introduced by the Visual Graphics Group (VGG) from the University of Oxford and are known for their simplicity and depth.

Architecture VGG models, particularly VGG16 and VGG19, consist of 16 and 19 layers, respectively. They use small (3x3) convolution filters throughout the entire network, increasing depth while controlling the number of parameters. These models are widely used as feature extractors for various computer vision tasks.

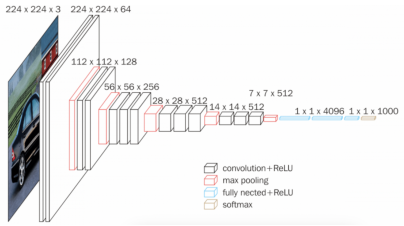

Figure 4: VGG Architecture.

Embedding Space The embedding space in VGG networks represents a high-dimensional vector space where each dimension captures distinct learned features of images. As an image progresses through the VGG's convolutional layers, it is transformed from raw pixel data into complex feature representations.

Early layers detect basic features such as edges, while deeper layers capture more abstract elements like parts of faces. The final convolutional or fully connected layers output an embedding—a compact vector that encodes the most critical aspects of the image.

Model Explanation VGGNet processes images of size 224x224 pixels, commonly extracting the central 224x224 patch from images for consistent input dimensions during the ImageNet competition. Its convolutional layers utilize a minimal 3x3 receptive field, capable of capturing basic directional movements, along with 1x1 filters for linear transformations followed by ReLU activation, which enhances training efficiency.

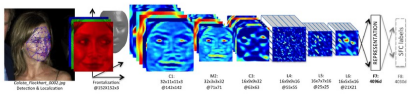

Figure 5: Working of VGG Architecture.

The convolution stride is fixed at 1 pixel to preserve spatial resolution. VGGNet's architecture includes hidden layers exclusively using ReLU and avoids Local Response Normalization (LRN) to reduce memory and training time without sacrificing accuracy. The network features three fully-connected layers, with the first two containing 4096 channels each and the third comprising 1000 channels for classifying into as many categories.

Support Vector Machines

Support Vector Machines (SVM) are supervised learning methods used for classification, regression, and outlier detection. They work by finding a hyperplane in an N-dimensional space (where N is the number of features) that maximally separates different classes.

Key Concepts and Operations

Mathematical Formulation

- Input data vectors x
- Class labels $y_i \in \{-1, 1\}$
- Weight vector w and bias b for the hyperplane

Kernel Trick Transforms the input space to a higher dimensional space using functions like:

- Linear: $K(x_i, x_j) = x_i \cdot x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d$
- Radial Basis Function (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$

Soft Margin and Slack Variables Allows some misclassifications using slack variables ξ_i:

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Here, C balances margin maximization with classification error.

SVC Algorithm

1. Select kernel, parameters, and regularization.
2. Transform data to kernel space.
3. Solve quadratic optimization for w and b.
4. Use support vectors and coefficients to define the decision function.

Training and Prediction Adjust w and b to minimize error on the training dataset; use the decision function to classify new data points.

Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is an artificial neural network used for solving classification and regression problems. It consists of an input layer, several hidden layers, and an output layer, with neurons in each layer connected to all neurons in the previous layer. Neurons use activation functions like sigmoid, tanh, or ReLU to introduce non-linearity, crucial for learning complex patterns.

Operation

MLP operation involves forward propagation, where input data passes through the network to produce output, and backpropagation, where the network adjusts its weights and biases to minimize error between its predictions and actual data. The network learns through repeated iterations (epochs), improving its prediction accuracy over time.

Key Processes

Forward Propagation Input data is processed layer-by-layer:

1. Compute weighted input for each layer l: $z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$, with $a^{(0)} = x$.
2. Apply activation function: $a^{(l)} = f^{(l)}(z^{(l)})$.

Backpropagation Adjust weights and biases based on the error:

1. Compute error at output: $\delta^{(L)} = \nabla_{a^{(L)}}\mathcal{L} \odot f'^{(L)}(z^{(L)})$.
2. Propagate error back and compute gradients for weights and biases.

Parameter Update Update parameters using gradient descent:

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

$$b^{(l)} = b^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial b^{(l)}}$$

Hyperparameter Tuning with Grid Search and Cross-Validation

Hyperparameter involves selecting the best combination of parameters for a model that governs the training process and can significantly impact the model's performance. We have performed hyperparameter tuning for SVM and MLP using a method called Grid Search along with Cross-Validation. A grid of hyperparameter values is defined, and the model is evaluated for each combination of these parameters.

For each combination, the model is trained multiple times on different partitions of the data through Cross-Validation (4-fold Cross-Validation).

Best Parameter Combination: Grid Search evaluates every possible combination of these hyperparameters and provides the combination of parameters that yields the best average performance across all validation sets.

Model Assessment: Grid Search also returns the performance metrics for each parameter combination, providing insights into how model performance varies with parameter changes. This includes an array of scores for each validation fold for each parameter set.

Data Augmentation

Data augmentation is a process of generating new training samples from the existing ones by applying random jitters and perturbations, thereby creating a more robust and diverse set of training data. The inclusion of data augmentation should increase the testing accuracy since the they model would be trained with images in different scenarios. The data augmentation techniques applied include rotations, translations, scaling, shearing, flipping, color perturbation and noise injection.

The data augmentation was applied both the models (SVM, MLP) produced lower test accuracy than the original score. This could be due to mismatch between training and test distributions or insufficient model complexity. If the augmented data diverges significantly from the distribution of the test set, the model may not generalize well, leading to poor test performance. The models may not have enough capacity to learn from the more complex, augmented dataset and thus fail to capture the additional variance introduced by the augmentations.

Results and Discussions

In our exploration of the FER dataset, we tested a diverse array of configurations to identify the most effective method for facial emotion recognition.

Experiment 1: Facenet reps with SVC Classification

1. Implementation Description

- Model Components: This model has embeddings from FaceNet. Classification is performed using SVM
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.6323104253021701
Training Precision: 0.6677122158963346
Training Recall: 0.60571252698334154
Training F1 Score: 0.6271997013066145
Test Accuracy: 0.5057118974644748
Test Precision: 0.5247477311485594
Test Recall: 0.46502215191964946
Test F1 Score: 0.48121490639740655
```

Figure 6: Facenet reps with SVC Classification: Metrics

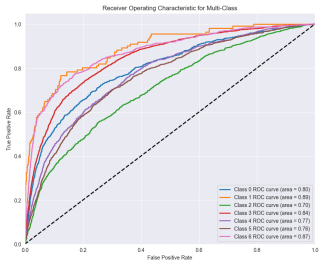

Figure 7: Facenet reps with SVC Classification: ROC

3. Observations and Potential Improvements

- Apply PCA to Evaluate Potential Improvements

Experiment 2: Facenet reps with SVC Classification Following PCA Implementation

1. Configuration Description

- Model Components: This model has embeddings from FaceNet. Classification is performed using SVM
- Preprocessing Steps: PCA and scaling has been performed.
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9985022118499425
Training Precision: 0.9982245740329072
Training Recall: 0.9981942884062736
Training F1 Score: 0.9982092581746134
Test Accuracy: 0.33212594037336307
Test Precision: 0.8377737535913324
Test Recall: 0.26958409354641494
Test F1 Score: 0.26981024521592045
```

Figure 8: Facenet reps with SVC Classification Following PCA Implementation: Metrics

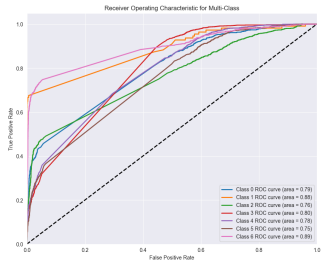

Figure 9: Facenet reps with SVC Classification Following PCA Implementation: ROC

3. Observations and Potential Improvements

- Perform Hyperparameter Tuning

Experiment 3: Facenet reps with SVC Classification with Best Params Following PCA Implementation

1. Configuration Description

- Model Components: This model has embeddings from FaceNet. Classification is performed using SVM
- Preprocessing Steps: PCA and scaling has been performed.
- Parameter Settings: Hyperparameter Tuning has been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9984325472848236
Training Precision: 0.9981547288684841
Training Recall: 0.9981041861402016
Training F1 Score: 0.9981293326210254
Test Accuracy: 0.33212594037336307
Test Precision: 0.8377737535913324
Test Recall: 0.26958409354641494
Test F1 Score: 0.26981024521592045
```

Figure 10: Facenet reps with SVC Classification (Best Params) Following PCA Implementation: metrics

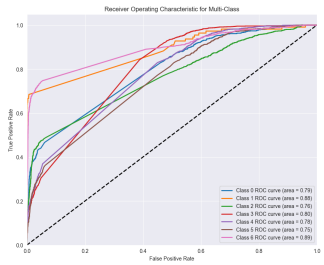

Figure 11: Facenet reps with SVC Classification (Best Params) Following PCA Implementation: ROC

3. Observations and Potential Improvements

- Perform Hyperparameter Tuning prior without doing PCA

Experiment 4: Facenet reps with SVC Classification with Best Params

1. Configuration Description

- Model Components: This model has embeddings from FaceNet. Classification is performed using SVM
- Preprocessing Steps: Scaling of the dataset. PCA was removed due to reduction in accuracy values.
- Parameter Settings: Hyperparameter Tuning has been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9852659444773416
Training Precision: 0.9862432989684814
Training Recall: 0.9843971920548594
Training F1 Score: 0.9853093625075651

Test Accuracy: 0.5792699916411257
Test Precision: 0.6045749597280335
Test Recall: 0.5821882404496537
Test F1 Score: 0.591811913091196
```

Figure 12: Facenet reps with SVC Classification (Best Params) : metrics

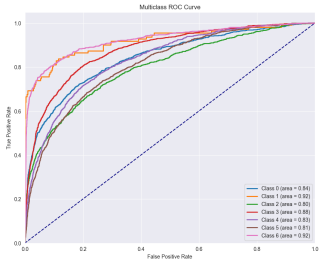

Figure 13: Facenet reps with SVC Classification (Best Params) : ROC

3. Observations and Potential Improvements

- This model has performed considerably well.

Experiment 5: VGG reps with SVC Classification

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using SVM
- Preprocessing Steps: Only scaling of the dataset has been performed
- Parameter Settings: Hyperparameter Tuning has not been performed for this configuration
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.996
Training Precision: 0.9936539798375411
Training Recall: 0.9933428345353603
Training F1 Score: 0.9934396787628101
Test Accuracy: 0.6673003802281369
Test Precision: 0.6442969371865984
Test Recall: 0.5992720970140105
Test F1 Score: 0.6169088328398322
```

Figure 14: VGG reps with SVC Classification: metrics

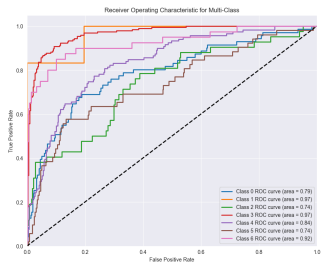

Figure 15: VGG reps with SVC Classification: ROC

3. Recommendations and Potential Improvements

- Apply PCA to Evaluate Potential Improvements

Experiment 6: VGG reps with SVC Classification Following PCA Implementation

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using SVM
- Preprocessing Steps: Scaling and PCA on the dataset
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9985
Training Precision: 0.9982498126908065
Training Recall: 0.9971821866231804
Training F1 Score: 0.9977087385520876
Test Accuracy: 0.42585551330798477
Test Precision: 0.7230046948356808
Test Recall: 0.3030062729317079
Test F1 Score: 0.3026786546786547
```

Figure 16: VGG reps with SVC Classification Following PCA Implementation: Metrics

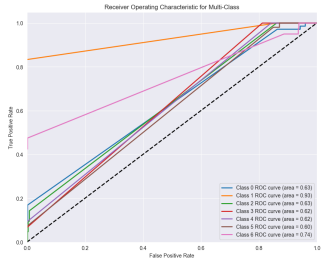

Figure 17: VGG reps with SVC Classification Following PCA Implementation: ROC

3. Recommendations and Potential Improvements

- Perform Hyperparameter Tuning

Experiment 7: VGG reps with SVC Classification with Best Params Following PCA Implementation

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using SVM
- Preprocessing Steps: Scaling and PCA on the dataset
- Parameter Settings: Hyperparameter Tuning has been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9985
Training Precision: 0.9982498126908065
Training Recall: 0.9971821866231804
Training F1 Score: 0.9977087385520876
Test Accuracy: 0.4239543726235741 6
Test Precision: 0.7228915662650602
Test Recall: 0.3009942004971003
Test F1 Score: 0.2989071328667871
```

Figure 18: VGG reps with SVC Classification (Best Params) Following PCA Implementation: metrics

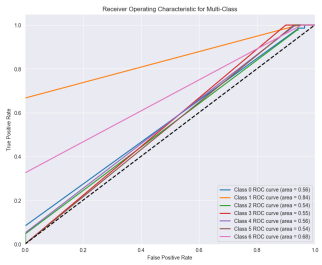

Figure 19: VGG reps with SVC Classification (Best Params) Following PCA Implementation: ROC

3. Recommendations and Potential Improvements

- Perform Hyperparameter Tuning prior to PCA

Experiment 8: VGG reps with SVC Classification with Best Params

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using SVM
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9965
Training Precision: 0.9951521933938269
Training Recall: 0.9936653111558054
Training F1 Score: 0.994334875480769

Test Accuracy: 0.6844106463878327
Test Precision: 0.6804100777502312
Test Recall: 0.6032354994311608
Test F1 Score: 0.6283099970894302
```

Figure 20: VGG reps with SVC Classification (Best Params) : metrics

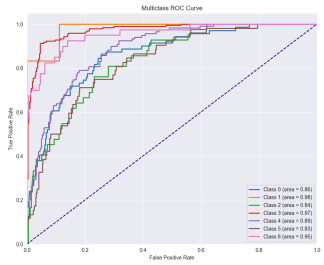

Figure 21: Facenet reps with SVC Classification (Best Params) : ROC

3. Recommendations and Potential Improvements

- This model has performed considerably well.

Experiment 9: FaceNet Reps with MLP Classification

1. Configuration Description

- Model Components: This model has embeddings from Facenet. Classification is performed using MLP
- Preprocessing Steps: Scaling of the dataset has been performed
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.5815249573304538
Training Precision: 0.5660059900954683
Training Recall: 0.5740960989429225
Training F1 Score: 0.5670473343364125
Test Accuracy: 0.4524937308442463
Test Precision: 0.41533418746100453
Test Recall: 0.4256132108046885
Test F1 Score: 0.4178519695518645
```

Figure 22: FaceNet reps with MLP Classification: metrics

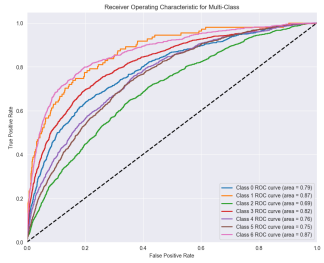

Figure 23: FaceNet reps with MLP Classification: ROC

3. Recommendations and Potential Improvements

- Perform HyperParameter Tuning

Experiment 10: VGG Reps with MLP Classification

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using MLP
- Preprocessing Steps: Scaling of the dataset has been performed
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: FER Dataset

2. Performance Evaluation

```
Training Accuracy: 0.9985
Training Precision: 0.9976063120473058
Training Recall: 0.9978249609679863
Training F1 Score: 0.9977149409156233
Test Accuracy: 0.6444866920152091
Test Precision: 0.5803584083374485
Test Recall: 0.5769676219040472
Test F1 Score: 0.5738079170063068
```

Figure 24: VGG reps with MLP Classification: metrics

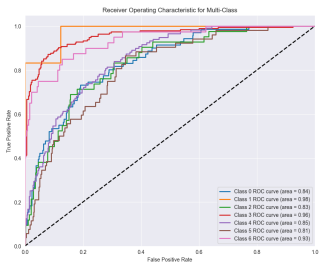

Figure 25: VGG reps with SVC Classification: ROC

3. Recommendations and Potential Improvements

- Perform Hyperparameter tuning.

Experiment 11: VGG Reps with MLP Classification (Best Params)

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using MLP
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning
- Dataset: FER

2. Performance Evaluation

```
Training Accuracy: 0.9985
Training Precision: 0.9982498126908065
Training Recall: 0.9971821866231804
Training F1 Score: 0.9977087385520876

Test Accuracy: 0.6520912547528517
Test Precision: 0.6314089749356155
Test Recall: 0.5884914033714399
Test F1 Score: 0.6046189240228909
```

Figure 26: VGG reps with MLP Classification(Best Params): metrics

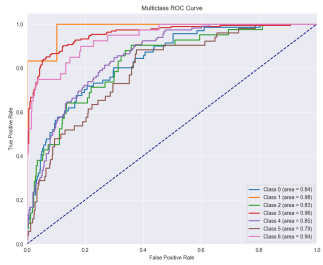

Figure 27: VGG reps with MLP Classification: ROC

5. Recommendations and Potential Improvements

- Perform HyperParameter Tuning

Experiment 12: Facenet Reps with MLP Classification (Best Params)

1. Configuration Description

- Model Components: This model has embeddings from FaceNet. Classification is performed using MLP
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning
- Dataset: FER

2. Performance Evaluation

- Metric Results:

```
Training Accuracy: 0.6986659235779721
Training Precision: 0.7190064372120993
Training Recall: 0.7110942779457938
Training F1 Score: 0.7123763589462069

Test Accuracy: 0.45695179715798273
Test Precision: 0.42873389963478176
Test Recall: 0.4479690912662839
Test F1 Score: 0.436093106924696
```

Figure 28: Facenet reps with MLP Classification(Best Params): metrics

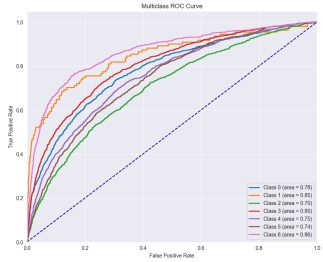

Figure 29: Facenet reps with MLP Classification: ROC

5. Recommendations and Potential Improvements

- Perform HyperParameter Tuning

Experiments with ExpW Dataset

Experiment 13: Facenet reps with SVC Classification

1. Configuration Description

- Model Components: This model has embeddings from Facenet. Classification is performed using SVM
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning has not been performed
- Dataset: ExpW

3. Performance Evaluation

```
Training Accuracy: 0.996
Training Precision: 0.9936539798375411
Training Recall: 0.9933428345353603
Training F1 Score: 0.9934396787628101
Test Accuracy: 0.6673003802281369
Test Precision: 0.6442969371865984
Test Recall: 0.5992720970140105
Test F1 Score: 0.6169088328398322
```

Figure 30: Facenet reps with SVC Classification: Metrics ExpW Dataset

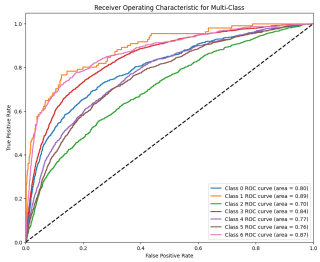

Figure 31: Facenet reps with SVC Classification: ROC ExpW Dataset

Experiment 14: VGG reps with SVC Classification

1. Configuration Description

- Model Components: This model has embeddings from VGG. Classification is performed using SVM
- Preprocessing Steps: Scaling of the dataset
- Parameter Settings: Hyperparameter Tuning
- Dataset: ExpW

2. Performance Evaluation

```
Training Accuracy: 0.996
Training Precision: 0.9936539798375411
Training Recall: 0.9933428345353603
Training F1 Score: 0.9934396787628101
Test Accuracy: 0.6673003802281369
Test Precision: 0.6442969371865984
Test Recall: 0.5992720970140105
Test F1 Score: 0.6169088328398322
```

Figure 32: VGG reps with SVC Classification: Metrics ExpW Dataset

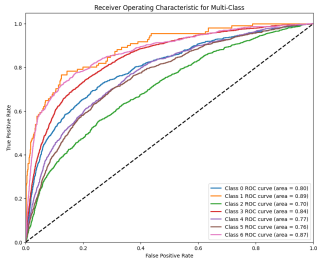

Figure 33: VGG reps with SVC Classification: ROC ExpW Dataset

We attempted a similar analysis using the Expressions in the Wild database, but the outcomes were unsatisfactory. Consequently, we opted for the FER dataset, which yielded better results.

Our analysis reveals a propensity for the classifier to confuse positive emotions with one another, similarly observed with negative emotions. This phenomenon is largely attributable to the nuanced similarities among facial expressions of positive emotions such as happiness and surprise. For example, both emotions may involve eye-widening. Similarly, expressions of happiness and neutrality can sometimes be indistinguishable, particularly in individuals whose facial structures make these emotions appear similar, leading to frequent misclassification.

Figure 34: Missclassifications

- **Image 1:** The classifier incorrectly predicted 'neutral' for a facial expression that was in fact 'happy.' The person's lips were relatively flat, without the pronounced upward curve typical of a smile, possibly leading the classifier to interpret the expression as neutral.
- **Image 2:** The classifier misidentified a 'happy' expression as 'surprised.' The widened eyes, a hallmark of surprise, may have overshadowed the more subtle signs of a smile, resulting in a skewed classification.
- **Image 3:** In this case, the expression was labeled 'happy' but classified as 'surprised,' likely due to the classifier misinterpreting the facial cues.
- **Image 4:** Again, an expression labeled 'happy' was classified as 'neutral.' The limited movement in the lips, which lacked the distinctive curvature of a smile, likely influenced this classification.

These findings indicate that some images in the dataset might be inaccurately labeled or that their features do not distinctly aid the classifier's learning, thereby blurring the decision boundaries between classes.

Conclusion

In this study, we evaluated various configurations of machine learning models for facial recognition tasks, employing architectures involving FaceNet and VGG-Face embeddings combined with different classifiers. Our findings indicate that the model combining PCA, VGG embeddings, SVM, and optimized parameters (C=10, gamma=1, kernel='rbf') yielded the highest accuracy. This suggests that VGG-Face provides more effective facial embeddings compared to FaceNet in our experiments.

We also explored the impact of PCA on model performance. While PCA helped in reducing model complexity by lowering dimensionality, it slightly decreased accuracy, likely due to the loss of relevant information. Through parameter tuning, we identified optimal settings for the SVM classifier, which significantly enhanced model performance. Data augmentation, another technique we tested to enhance model robustness, did not improve accuracy; instead, it led to reduced performance, potentially because of increased complexity overwhelming the model's learning capabilities. Our future endeavors should pivot towards exploring advanced iterations or alternatives to PCA. Delving into modified versions of PCA may unravel more suitable methods for dimensionality reduction that could enhance our model's ability to discriminate effectively.

Overall, the integration of VGG embeddings with SVM, enhanced through strategic parameter optimization, stands out as the most effective approach in our series of experiments for facial recognition tasks.

Looking forward, there's a scope to experiment beyond the confines of FaceNet and VGG architectures. There exists a plethora of other models that could potentially offer superior results, and our next steps could involve evaluating a broader spectrum of these options.

References

[1] Sergey Ioffe, Christian Szegedy, FaceNet: A Unified Embedding for Face Recognition and Clustering, *arXiv preprint arXiv:1503.03832*, 2015, https://arxiv.org/abs/1503.03832.

[2] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, *arXiv preprint arXiv:1409.1556v6*, 2014, https://arxiv.org/abs/1409.1556v6.

[3] Keyur Patel et al., Facial Sentiment Analysis Using AI Techniques: State-of-the-Art, Taxonomies, and Challenges, *IEEE Access*, vol. 8, pp. 90495-90519, 2020, https://doi.org/10.1109/ACCESS.2020.2993803, Keywords: Feature extraction; Sentiment analysis; Taxonomy; Machine learning; Face recognition; Face; Facial sentiment analysis; machine learning; deep learning; convolutional neural network; deep belief network; artificial intelligence.

[4] Author(s), Sentimental Analysis of People Using Facial Expression, *[Journal/Conference]*, year, https://www.researchgate.net/publication/371493085_Sentimental_Analysis_of_People_Using_Facial_Expression.