Image Classification: Emotion Recognition

Mounika Seelam
811344833
Dept. of Computer Science
Kent State University
Ohio, USA
mseelam2@kent.edu

Prajwal Devaraj
811351381
Dept. of Computer Science
Kent State University
Ohio, USA
pdevaraj@kent.edu

Bhanu Prasad Dharavathu
811349718
Dept. of Computer Science
Kent State University
Ohio, USA
bdharav1@kent.edu

***Abstract*— This project report details the design, implementation, and performance optimization of an image classification model utilizing Convolutional Neural Networks (CNNs). The main aim was to categorize images into separate classes using deep learning methods. We began with basic CNN architecture based on publicly available code and aimed to enhance its performance by implementing various optimization strategies. The model has a few convolutional layers, pooling layers, and fully connected layers that interact with one another to learn and extract hierarchical feature representations of the input images. The model was trained using a large, labeled dataset, hence can learn the required features and patterns for effective classification. Primarily, the following enhancements were performed using data augmentation, adjustment of hyperparameters, early stopping, and use of regularization methods. These improvements led to the significant increases in training and validation accuracy of the model. The effectiveness of the CNN in image classification was determined based on its accuracy in classifying new images in the test set, showing the ability of CNNs in this classification. The results highlight the importance of iterative optimization for deep learning models and suggest possible future work in further optimizing the model and its real-world applications.**

Keywords—Facial Emotion Recognition (FER), Deep Learning, Convolutional Neural Networks (CNN), Data Augmentation, Class Imbalance, Overfitting, Regularization, Image Classification, Model Architecture, Image Analysis

I. INTRODUCTION

In this section, we provide an overview of the project, discussing the fundamental concepts behind Convolutional Neural Networks (CNNs), the goals of the project, and the methodology we employed to enhance the performance of the image classification model. The following subsections detail the core aspects of the project:

A. Overview of Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of deep learning models that have worked incredibly well in all domains, particularly in image classification. CNNs are specifically designed to process data that has a grid-like topology, i.e., images. The primary advantage of CNNs over other machine learning algorithms is that they are capable of learning automatically and adaptively spatial hierarchies of features from the input data. This allows CNNs to be able to effectively detect patterns in images, and thus they are extremely valuable for computer vision tasks, e.g., object detection, image segmentation, and classification.

The architecture of a typical CNN consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. All these are integrated to acquire representative features from the input images and enable the model to predict or classify according to the features.

B. Problem Statement and Project Objective

The main objective of this project is to design and implement a CNN-based image classification model that can classify images into distinct categories with high accuracy. The model should be capable of learning from a large, labelled dataset and generalizing well to unseen images.

To achieve this, we aim to:

1. Build a CNN model
2. Explore and implement techniques to improve the model's performance, such as data augmentation, hyperparameter tuning, and regularization.
3. Evaluate the model's accuracy and performance using a separate test dataset.

C. Structure of Convolutional Neural Networks

CNNs consist of several elementary building blocks that work together to learn and extract features from the input data. They are:

- Convolutional Layers: The elementary building block of a CNN. Filters (or kernels) are applied to the input data, extracting local features through a sliding window approach.
- Pooling Layers: These layers reduce the spatial size of the data, hence making the model less sensitive to small translations or distortions in the input.
- Fully Connected Layers: These typically come at the end of the CNN architecture, and they connect each neuron to each other, combining the learned features to perform the final classification.

The scope of this project is focused on implementing a CNN model for image classification tasks. Although the model is trained on a labeled dataset of images, the primary challenge lies in improving its generalization ability to unseen data. We aimed to design an efficient CNN that can be used for image classification problems in various domains, such as facial recognition.

II. RELATED WORKS

A. Foundational Implementation

Our work builds upon the facial emotion recognition implementation of Hackers Realm [1], which demonstrated a minimalist CNN architecture to classify seven emotional states (anger, disgust, fear, happiness, neutrality, sadness, and surprise). Their approach utilized a sequential CNN model

with minimal regularization and achieved mediocre performance on benchmark emotion datasets. The reference had an training accuracy of 71% and validation accuracy of 63%. While it was helpful as a baseline, their implementation did not handle class imbalance and overfitting, particularly for underrepresented emotion classes like disgust and fear.

B. *CNN Architectures for Emotion Recognition*

Convolutional Neural Networks have emerged as the dominant approach to facial emotion recognition due to their ability to learn to automatically extract spatial features from face images. Mollahosseini et al. [2] proposed deeper network models incorporating inception modules and reported significantly improved performance over shallow networks. Li and Deng [3] have provided a comprehensive survey of deep learning approaches to FER, highlighting the shift from simple architectures to more sophisticated ones with residual connections and attention mechanisms.

C. *Addressing Dataset Challenges*

Facial emotion data are subject to several challenges like data quality problems and class imbalance. Wang et al. [4] proposed the Self-Cure Network for reducing uncertainty in facial expression data by self-attention based weighing of training samples and a relabelling strategy for difficult samples. Our work addresses this in a complementary manner by solely addressing the problem of class imbalance through selective data augmentation of the minority classes of emotion to balance the training set.

Zhu et al. [5] offered effective techniques for biasing minority classes in emotion recognition databases and showed that domain-specific transformations can significantly improve the recognition performance for such expressions as disgust and fear, which are typically underrepresented.

D. *Regularization and Optimization Strategies*

To promote generalization, recent FER implementations have incorporated a number of regularization techniques. The application of dropout, batch normalization, and L2 regularization has been particularly successful, as demonstrated by Zhang et al. [6]. Our model incorporates these techniques with careful parameter tuning to deter overfitting without degrading discriminative power.

Adaptive learning rate scheduling has also been a central component to train effective FER models. The Reduce-on-Plateau method employed in our model follows best practices from Loshchilov and Hutter [7] in that it enables increasingly finer-grained optimization during training.

E. *Recent Advances*

While our implementation focuses on optimizing a CNN-based approach, some recent research has explored different alternative architectures. Attention mechanisms based on discriminative facial regions have proven promising in Li et al. [8], and transformer-inspired architectures based on NLP have begun to appear in state-of-the-art FER systems [9]. These approaches are possible directions for future work to build upon our research.

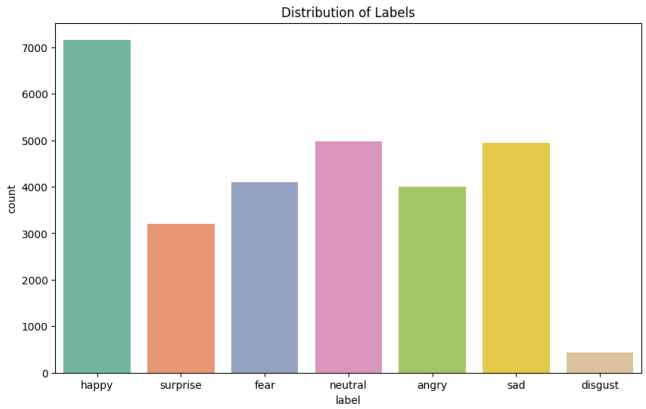
III. METHODOLOGY

This section outlines the methodology employed for developing a facial expression recognition system. The process includes data collection and preprocessing, model architecture design, hyperparameter tuning, training procedures, and performance improvement techniques. This comprehensive approach ensures the robustness and accuracy of the final model.

A. *Data Collection and Preprocessing*

The dataset we used in this project was the Face Expression Recognition dataset obtained from Kaggle [10], which consists of images of faces, each labeled with one of seven basic emotions: angry, disgust, fear, happy, neutral, sad, and surprise. The original dataset was divided into training and validation directories, each containing subdirectories for the various emotion categories. This structure was reorganized into Pandas DataFrames to facilitate data manipulation, with columns created for image paths and corresponding labels. The data was converted into Pandas DataFrames using a custom loadDataset function to streamline data access and manipulation.

Fig. 1. Original Training Data Distribution



The class distribution in the original training dataset was as follows (Fig. 1):

- happy: 7164
- neutral: 4982
- sad: 4938
- fear: 4103
- angry: 3993
- surprise: 3205
- disgust: 436

To ensure consistent input for the model, a preprocessing pipeline was established. This pipeline included several key steps:

- Resizing: All images were resized to a uniform resolution of 48x48 pixels using the resize_image function, employing Image.LANCZOS resampling to maintain image quality during downscaling.
- Normalization: Pixel values were normalized to the range of [0, 1] by dividing by 255, which aids in faster convergence during training.
- Feature Extraction: The extractFeatures function was used to preprocess each image, converting them into NumPy arrays and reshaping them into a 48×48×1 format suitable for CNN input.

B. *Model Architecture*

The chosen model architecture was a Convolutional Neural Network (CNN). The CNN architecture comprised of multiple convolutional layers (Conv2D), batch normalization layers (BatchNormalization), max-pooling layers (MaxPooling2D), and dropout layers (Dropout). L2 regularization was applied to the convolutional layers to prevent overfitting. The specific layers were:

1. Convolutional Layer 1: 32 filters, kernel size (3, 3), ReLU activation, L2 regularization with a parameter of 0.001, and batch normalization. This was followed by a max-pooling layer with a pool size of (2, 2) and a dropout layer with a rate of 0.4.
2. Convolutional Layer 2: 64 filters, kernel size (3, 3), ReLU activation, L2 regularization with a parameter of 0.01, and batch normalization. This was followed by a max-pooling layer with a pool size of (2, 2) and a dropout layer with a rate of 0.4.
3. Convolutional Layer 3: 128 filters, kernel size (3, 3), ReLU activation, L2 regularization with a parameter of 0.01, and batch normalization. This was followed by a max-pooling layer with a pool size of (2, 2) and a dropout layer with a rate of 0.4.
4. Convolutional Layer 4: 512 filters, kernel size (3, 3), ReLU activation, L2 regularization with a parameter of 0.01, and batch normalization. This was followed by a max-pooling layer with a pool size of (2, 2) and a dropout layer with a rate of 0.4.
5. Fully Connected Layers: A flatten layer, a dense layer with 256 units and ReLU activation, and L2 regularization with a parameter of 0.01, followed by a dropout layer with a rate of 0.5. The architecture concludes with a final dense layer consisting of 7 units employing a softmax activation function, corresponding to the seven emotion classes.

We chose ReLU activation function for its efficiency in training convolutional layers and deep learning models. The final dense layer used a softmax activation function to output the predicted probabilities for each emotion class.

C. *Hyperparameter Tuning and Training*

To optimize the model's performance, we have carefully tuned several hyperparameters. These included:

- Learning Rate: Set to 0.0001.
- Batch Size: Set to 64.
- Epochs: The model was trained for a maximum of 500 epochs.
- Dropout Rate: A dropout rate of 0.4 was used after each max pooling layer to prevent overfitting.

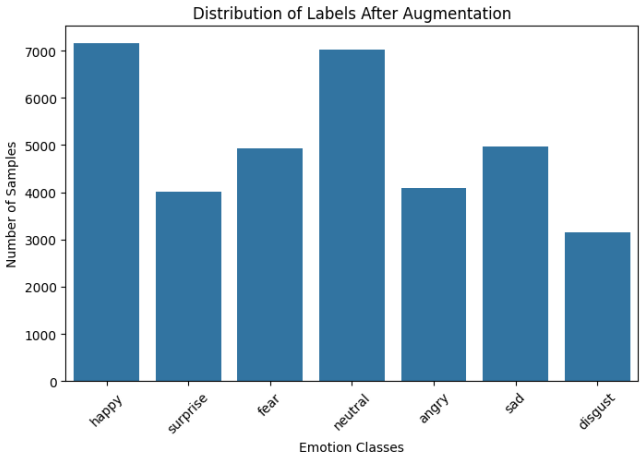
We have selected Adam optimizer, and the model was compiled using the categorical cross-entropy loss function, appropriate for multi-class classification, with accuracy as the evaluation metric. Class weights were computed using compute_class_weight from Scikit-learn to address the class imbalance issue. These weights were then passed to the fit method during training to give more importance to the underrepresented classes.

D. *Performance Improvement Techniques*

To further enhance the model's performance, we employed several strategies:

- **Data Augmentation:** To address the issue of class imbalance and improve the model's generalization ability, data augmentation techniques were employed. After improvement, the class distribution was aimed to be around:
  - happy: 7164
  - neutral: 7000
  - sad: 5000
  - fear: 4000
  - angry: 4100
  - surprise: 4000
  - disgust: 3500

Fig. 2.
Class Distribution after Augmentation



We used the ImageDataGenerator class from TensorFlow Keras to apply a series of random transformations to the existing images, generating new, synthetic training examples. The transformations included a rotation range of 30 degrees, width and height shifts of 0.2, a shear range of 0.2, zoom range of 0.2, horizontal flips, and brightness adjustments between 0.8 and 1.2. The augment_classes function was defined to apply these augmentations specifically to the under-represented classes, ensuring that each class had a similar number of training examples. The number of augmented images generated for each underrepresented class was calculated to match the sample count of the largest class, "happy", which had 7000 samples (Fig. 1).

- **ReduceLROnPlateau Callback:** The ReduceLROnPlateau callback was employed to reduce the learning rate by a factor of 0.75 when the validation loss plateaued for eight epochs, with a minimum learning rate of 0.00001.
- **EarlyStopping Callback:** The EarlyStopping callback was configured to monitor the validation loss and halt training if the loss did not improve for 20 epochs, restoring the best weights observed during training.
- **ModelCheckpoint Callback:** The ModelCheckpoint callback was used to save the model with the best validation loss.

By using these techniques, we aimed to prevent overfitting, improve model generalization, and optimize training efficiency.

E. Evaluation Metrics

We have assessed the model performance using the validation dataset, and the training history, including accuracy and loss, was visualized to analyze learning and identify potential overfitting. A confusion matrix was generated to provide detailed insights into the classification performance of each emotion category.

IV. Result

A. Model Performance Overview

Our facial expression recognition model, built with a four-layer CNN architecture with regularization techniques, was trained and evaluated on a dataset of seven emotions: angry, disgust, fear, happy, neutral, sad, and surprise. The model achieved a training accuracy of 81.51% and a validation accuracy of 66.12% at the optimal stopping point (Epoch 219), demonstrating reasonable performance but with clear areas for improvement.

B. Training and Validation Metrics

a) Learning Progression

The training curves (Fig. 3*a* and Fig. 3b) reveal important insights into the model's dynamics of learning:

- **Initial Learning Phase:** Both training and validation accuracy improved rapidly during the first 50 epochs, reaching approximately 60% and 65% respectively.
- **Middle Learning Phase:** Between epochs 50-150, the training accuracy continued to climb steadily from 65% to 75%, while validation accuracy plateaued around 65-66%.
- **Late Learning Phase:** After epoch 150, the training accuracy continued to increase (reaching 81.51%), but the validation accuracy remained relatively constant, indicating the onset of overfitting despite our implemented regularization techniques.

b) Accuracy vs. Loss Analysis

The divergence between training and validation accuracy (81.51% vs. 66.12%) suggests the model was becoming increasingly specialized to the training data. This is further confirmed by examining the loss curves (Fig. 3b):

- The training loss decreased consistently throughout the training, reaching 0.6952 at Epoch 219.
- Validation loss initially decreased alongside training loss but stabilized around 1.1795.
- The growing gap between training and validation loss confirms the overfitting hypothesis, suggesting that even our dropout rates of 0.4-0.5 and L2 regularization (0.01) were not sufficient to fully prevent overfitting.

It's worth noting here that the early stopping mechanism functioned as intended, halting training at epoch 239 (with best weights from epoch 219) before overfitting could become more severe.

Fig. 3.

Fig. 1a. Training and Validation Accuracy

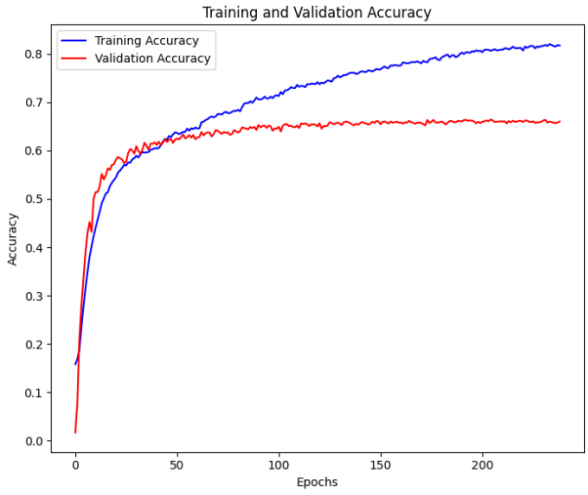


Fig. 1b. Training and Validation Loss

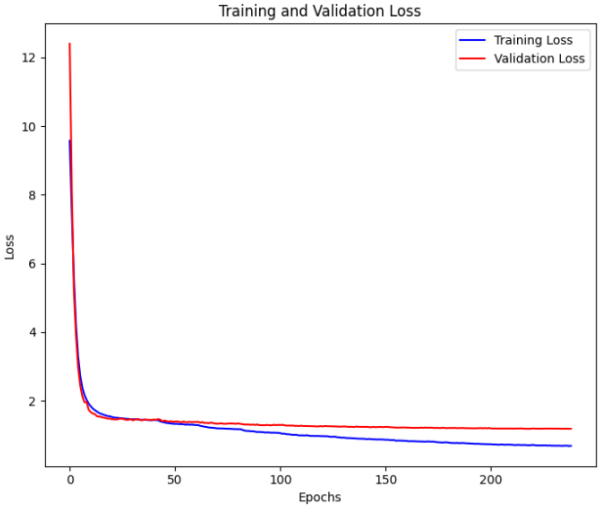


Fig. 4.

Confusion Matrix

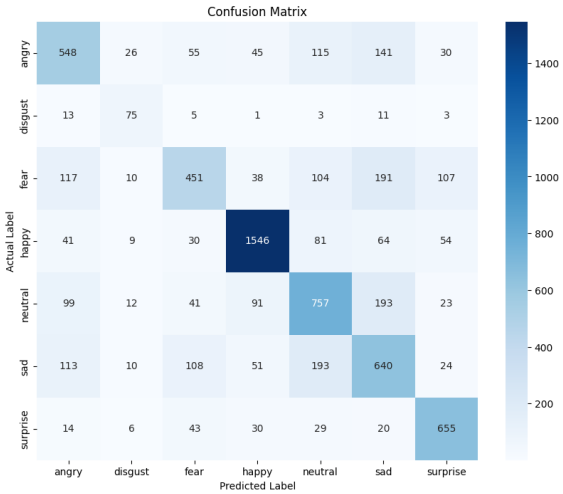


Fig. 5.
Classification Report

```
Classification Report:
              precision    recall  f1-score   support

       angry       0.58      0.57      0.58       960
     disgust       0.51      0.68      0.58       111
        fear       0.62      0.44      0.52      1018
       happy       0.86      0.85      0.85      1825
     neutral       0.59      0.62      0.61      1216
         sad       0.51      0.56      0.53      1139
    surprise       0.73      0.82      0.77       797

    accuracy                           0.66      7066
   macro avg       0.63      0.65      0.63      7066
weighted avg       0.66      0.66      0.66      7066
```

C. *Classification Performance by Emotion*

a) High-Performance Categories

- Happy: Achieved the highest recognition rate with 1546 correct classifications out of 1825 instances (84.7% accuracy).
- Surprise: Performed well with 655 correct classifications out of 797 instances (82.2% accuracy).

These categories performed best due to their distinctive facial features, such as smiles for happiness and wide eyes for surprise, which are more universally expressed across different individuals. Additionally, the "happy" class had the highest representation in our original dataset (7164 samples), which likely contributed to its strong performance (Fig. 5).

b) Moderate-Performance Categories

- Neutral: Achieved 757 correct classifications out of 1216 instances (62.3% accuracy)
- Sad: Achieved 640 correct classifications out of 1139 instances (56.2% accuracy)
- Angry: Achieved 548 correct classifications out of 960 instances (57.1% accuracy)
- Fear: Achieved 451 correct classifications out of 1018 instances (44.3% accuracy)

These moderate-performance categories reflect the inherent challenges in distinguishing between certain emotional expressions, particularly those with subtle differences (Fig. 5).

D. *Challenging Categories*

- Disgust: The most difficult emotion to classify, with only 75 correct classifications out of 111 instances (67.6% accuracy), though the small sample size affects the reliability of this metric (Fig. 5).

Despite our data augmentation efforts to increase the "disgusting" class from 436 to approximately 3500 samples, this category remained challenging. This suggests that synthetic augmentation alone may not fully compensate for limited original training examples.

E. *Error Analysis*

a) Confusion Patterns

Analyzing the confusion matrix (Fig. 4) reveals several notable patterns:

1. Fear-Sad Confusion: 191 instances of fear were misclassified as sad, representing the highest cross-category confusion.
2. Angry-Neutral Confusion: 115 angry expressions were misclassified as neutral.
3. Sad-Neutral Confusion: 193 sad expressions were misclassified as neutral, and 193 neutral expressions were misclassified as sad.
4. Fear-Surprise Confusion: 107 fear expressions were misclassified as surprise.

These confusion patterns align with psychological research showing that certain emotion pairs (fear/surprise, anger/disgust, sadness/neutral) share similar facial muscle activations, making them inherently more difficult to distinguish.

V. DISCUSSION

A. *Implementation Challenges and Solutions*

a) Data Imbalance Issues

One of the primary challenges encountered was class imbalance in the training dataset:

- "Happy" was overrepresented (7164 samples)
- "Disgust" was severely underrepresented (436 samples)
- Other categories ranged from 3205 to 4982 samples

Solutions implemented:

- Applied class weights computed using compute_class_weight from Scikit-learn during model training.
- Implemented extensive data augmentation targeting underrepresented classes.
- Achieved more balanced class distribution: happy (7164), neutral (7000), sad (5000), fear (4000), angry (4100), surprise (4000), and disgust (3500).

The augmentation strategy included rotation (±30°), width and height shifts (0.2), shear range (0.2), zoom range (0.2), horizontal flips, and brightness adjustments (0.8-1.2), which effectively expanded our training dataset while preserving the essential characteristics of each emotion.

b) Overfitting Concerns

As evidenced by the growing gap between training and validation metrics, overfitting emerged as a significant challenge despite our regularization efforts.

Solutions implemented:

- Added dropout layers after each max-pooling layer (rate of 0.4) and after the first dense layer (rate of 0.5).
- Applied L2 regularization to convolutional layers (parameter values of 0.001 to 0.01)
- Implemented batch normalization after each convolutional layer.
- Used early stopping (patience=20) which triggered at epoch 239.

- Employed the ReduceLROnPlateau callback to reduce the learning rate by a factor of 0.75 when validation loss plateaued for 8 epochs.

While these measures improved generalization, the persistent gap between training and validation performance suggests that more aggressive regularization might be beneficial in future iterations.

B. Architectural Refinement

The final CNN architecture represents the culmination of several refinements:

1. **Layer Depth Optimization:** We determined that four convolutional layers provided the best balance between model complexity and performance, with increasing filter counts ($32 \rightarrow 64 \rightarrow 128 \rightarrow 512$) to capture increasingly abstract features.
2. **Regularization Tuning:** L2 regularization parameters increased from 0.001 to 0.01 in deeper layers to provide stronger regularization where overfitting was more likely.
3. **Activation Function Selection:** ReLU activation was selected for all hidden layers due to its computational efficiency and effectiveness in preventing the vanishing gradient problem.
4. **Batch Normalization Integration:** Batch normalization was added after each convolutional layer to stabilize training and potentially allow for higher learning rates.

C. Performance Improvement Techniques

a) Hyperparameter Optimization

Finding the optimal combination of hyperparameters was crucial for maximizing model performance:

- Learning Rate: After experimentation, 0.0001 provided the best balance between convergence speed and stability.
- Batch Size: A batch size of 64 was selected as it offered a good trade-off between training speed and gradient accuracy.
- Optimizer Selection: The Adam optimizer was chosen for its adaptive learning rate capabilities and robustness to hyperparameter settings.

b) Training Strategies

We employed several training strategies to improve model convergence and final performance:

- Dynamic Learning Rate: The ReduceLROnPlateau callback reduced the learning rate when training plateaued, allowing for finer optimization as training progressed.
- Model Checkpointing: The best model weights were saved based on validation loss, ensuring that temporary fluctuations didn't impact final model selection.
- Training Duration Control: The maximum number of epochs was set to 500, but early stopping typically triggered much earlier (around epoch 239), preventing unnecessary computation and potential overfitting.

D. Computational Environment

We used Google Colab as our main development environment for this project. This cloud-based platform offered our deep learning application several important benefits. Our model training was greatly accelerated by Google Colab's free GPU access (NVIDIA T4 or P100), which cut down on the hours of work on local CPUs to minutes. We were able to iterate through several model topologies and hyperparameter setups more quickly as a result, which ultimately produced a final model that was more efficient. Furthermore, configuration costs and environment setup problems were removed by Colab's pre-installed deep learning libraries (TensorFlow, Keras) and smooth connection with Google Drive for data storage. Colab notebooks' collaborative features made it easier to work as a team in real time by enabling everyone to view, change, and run code at the same time. This was very helpful for troubleshooting and group development. Our ability to try out complex CNN architectures and large amounts of data enhancement would have been greatly limited without these computing resources.

E. Team Member Contributions

Our project was successfully completed through the collaborative efforts of three team members, each bringing unique skills and perspectives:

a) Prajwal Devraj: Data Processing and Model Architecture

- Developed the data preprocessing pipeline, including image resizing, normalization, and feature extraction
- Designed and implemented the CNN architecture with four convolutional layers
- Conducted literature review to identify best practices in facial emotion recognition
- Performed pre

b) Mounika Seelam: Training Optimization, Regularization and Report Preparation

- Implemented data augmentation techniques to address class imbalance
- Designed the regularization strategy, including dropout, L2 regularization, and batch normalization
- Set up the training infrastructure with callbacks for early stopping, learning rate reduction, and model checkpointing
- Conducted hyperparameter tuning to optimize model performance
- Contributed to the preparation of the final project report

c) Bhanu Prasad Dharavathu: Evaluation, Analysis and Report Preparation

- Designed and implemented the evaluation framework, including confusion matrix generation
- Analyzed model performance across different emotion categories
- Created visualizations for training/validation metrics and classification results

- Documented findings and prepared the final project report
- Identified limitations and proposed directions for future improvement

VI. FUTURE WORK

Based on our findings, several strategies could potentially improve model performance:

1. **Advanced Architectures:** Exploring deeper architectures or attention mechanisms could enhance feature extraction capabilities [11]. A facial expression recognition model based on an attention mechanism introduces a self-attention mechanism based on the residual network to overcome the limitation of the local operation of convolution operation, which improves the ability of the model to capture long-range associated features [12]. One study demonstrated that by adding attention mechanism, some important features, such as the eyes and mouth, are given more weight, thus improving recognition [13].
2. **Transfer Learning:** Leveraging pre-trained models on larger, high-resolution facial datasets might provide better initialization for fine-tuning. Transfer learning allows a model, pre-trained on enough data, to be used for a specific task where there is much less data [14].
3. **Ensemble Approaches:** Combining multiple models specialized for different emotion subsets might improve overall accuracy [15].
4. **Multi-modal Analysis:** Incorporating additional modalities (e.g., body posture, voice tone) could provide complementary information for emotion recognition [16].
5. **Temporal Information:** Extending to video-based recognition using recurrent neural networks could capture dynamic aspects of emotional expressions. In 2016, Fan et al. combined recursive neural network and C3D convolutional neural network for dynamic video emotion recognition, and the accuracy reached 59.02% on AFEW dataset [17]. In 2021, Liu proposed a network of Capsule-LSTM to better extract the time sequence information about video sequences for video facial expressions recognition, and the accuracy reached 40.16% on AFEW dataset [17].

VII. CONCLUSION

Our facial expression recognition project has successfully developed and evaluated a four-layer CNN model for classifying seven distinct emotional expressions from facial images. The model achieved a training accuracy of 81.51% and a validation accuracy of 66.12%, representing a reasonable performance given the inherent challenges in emotion recognition and the limitations of the dataset.

The developed model demonstrated varying performance across emotion categories, with "happy" and "surprise" expressions recognized with high accuracy (84.7% and 82.2% respectively), while more subtle emotions like "fear" proved challenging (44.3% accuracy). This pattern aligns with existing research in facial expression recognition, where distinctive emotions with clear visual markers are typically easier to classify than those with more nuanced expressions.

Our analysis of misclassifications revealed consistent patterns of confusion between visually similar emotional expressions, particularly between fear-sad, sad-neutral, and angry-neutral pairs. These findings highlight the fundamental challenge in emotion recognition: the inherent similarity and subjectivity in how emotions are expressed and perceived.

Despite implementing various optimization techniques, including dropout regularization, batch normalization, data augmentation, and early stopping, the persistent gap between training and validation performance indicates that emotion recognition from low-resolution static images has intrinsic limitations. The subjective nature of emotion labeling combined with the loss of subtle facial details in 48×48-pixel images created a performance ceiling that our model approached but could not exceed.

While not achieving human-level accuracy, our model demonstrates sufficient performance for applications where approximate emotion recognition adds value, such as:

- User experience research and feedback collection
- Basic sentiment analysis in educational technology
- Preliminary screening in mental health applications
- Entertainment and interactive media experiences

The model's stronger performance on certain emotion categories suggests that specialized implementations focusing on distinguishing between specific emotion subsets could yield more reliable results for targeted applications.

A. Learning Outcomes

This project provided practical experience with the complete machine learning development cycle. The challenges encountered in addressing overfitting and class imbalance enhanced our understanding of deep learning fundamentals. Our collaborative approach, with team members focusing on different aspects of the project, proved effective in tackling this complex problem, while Google Colab's computational resources enabled efficient experimentation with our model architecture.

In conclusion, while achieving perfect emotion recognition remains an open challenge, our project demonstrates that even relatively simple CNN architectures can capture meaningful patterns in facial expressions. Our findings contribute to the ongoing evolution of computational emotion recognition by identifying specific challenges in this domain, while the experience gained throughout this project has deepened our understanding of both the technical and collaborative aspects of machine learning development.

REFERENCES

[1] Hackers Realm, "Facial Emotion Recognition Using Python," 2023. *(https://www.hackersrealm.net/post/facial-emotion-recognition-using-python)*

[2] A. Mollahosseini, D. Chan and M. H. Mahoor, "Going deeper in facial expression recognition using deep neural networks," *2016 IEEE Winter*

Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 2016, pp. 1-10, doi: 10.1109/WACV.2016.7477450.

[3] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," in *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1195-1215, 1 July-Sept. 2022, doi: 10.1109/TAFFC.2020.2981446.

[4] Wang, K., Peng, X., Yang, J., Lu, S., & Qiao, Y. (2020). Suppressing uncertainties for large-scale facial expression recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6897-6906).

[5] W. Huang, S. Zhang, P. Zhang, Y. Zha, Y. Fang and Y. Zhang, "Identity-Aware Facial Expression Recognition Via Deep Metric Learning Based on Synthesized Images," in *IEEE Transactions on Multimedia*, vol. 24, pp. 3327-3339, 2022, doi: 10.1109/TMM.2021.3096068.

[6] Zhang, Ting. (2018). Facial Expression Recognition Based on Deep Learning: A Survey. 345-352. 10.1007/978-3-319-69096-4_48.

[7] Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983. https://arxiv.org/abs/1608.03983.

[8] Ling, H., Wu, J., Huang, J. et al. Attention-based convolutional neural network for deep face recognition. Multimed Tools Appl 79, 5595–5616 (2020). https://doi.org/10.1007/s11042-019-08422-2

[9] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

[10] https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset/data

[11] Improved facial emotion recognition model based on a novel deep ... (2024). https://www.nature.com/articles/s41598-024-79167-8

[12] Facial Expression Recognition Based on Attention Mechanism. (2025). https://onlinelibrary.wiley.com/doi/10.1155/2021/6624251

[13] CNN combined with attention module for facial expression recognition. (n.d.). https://iopscience.iop.org/article/10.1088/1742-6596/2646/1/012023

[14] [PDF] Transfer Learning for Facial Emotion Recognition on Small Datasets. (n.d.). https://www.iiis.org/CDs2024/CD2024Spring/papers/ZA778SL.pdf

[15] Xception CNN-Ensemble Learning Based Facial Emotion Recognition. (n.d.). https://ieeexplore.ieee.org/document/10011111/

[16] Investigating Emotional Body Posture Recognition in Adolescents ... (2021). https://pmc.ncbi.nlm.nih.gov/articles/PMC8154802/

[17] Research on real-time teachers' facial expression recognition based ... (2023). https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-023-01019-w