Emotion Recognition through Facial Expressions Using Advanced Neural Networks

Technical Report

Submitted by:

One Person Group A - Arifa Kokab

AAI-521: Applied Computer Vision

Instructor:

Professor Saeed Sardari

University of San Diego

Shiley-Marcos School of Engineering

Date:

December 9, 2024

TABLE OF CONTENTS

ABSTRACT

This report presents a comprehensive analysis of three deep learning models for facial emotion recognition: ResNet50 (Model 1), an Attention-Based Model with CBAM Blocks (Model 2), and a Fine-Tuned MobileNetV2 (Model 3). Using the FER-2013 dataset, we aimed to identify an optimal model that balances accuracy, computational efficiency, and generalization. Each model's architecture, preprocessing requirements, performance metrics, and challenges were analyzed in detail. The findings indicate that Model 3 achieved the highest performance, showcasing its suitability for the given task despite dataset-specific constraints.

INTRODUCTION

Facial emotion recognition (FER) is a complex and significant challenge in the field of computer vision, focusing on the classification of human emotions based on facial expressions. The ability to accurately identify and categorize emotions holds transformative potential across various applications, such as mental health monitoring, human-computer interaction, and neuromarketing. These fields rely on FER systems to interpret subtle emotional cues, enabling innovations like adaptive user interfaces, emotional well-being assessments, and consumer behavior analysis.

This project aimed to develop a robust FER system capable of classifying facial expressions into seven distinct emotional categories: *Angry*, *Disgust*, *Fear*, *Happy*, *Neutral*, *Sad*, and *Surprise*. Using the FER-2013 dataset, a benchmark resource consisting of grayscale images of facial expressions, the project evaluated three machine learning models—ResNet50, an Attention-Based Model, and a Fine-Tuned MobileNetV2. This report documents the iterative process of designing, training, and evaluating these models while addressing the challenges posed by the FER-2013 dataset, including low resolution, class imbalance, and noise in the data. Through meticulous preprocessing, model selection, and hyperparameter optimization, the study aimed to create an efficient and accurate emotion recognition system.

PROBLEM DEFINITION

Importance of Facial Emotion Recognition

Facial expressions are a universal form of human communication, transcending linguistic and cultural barriers. Recognizing emotions from facial expressions has far-reaching implications across various domains, such as:

- **Healthcare**: FER systems can assist clinicians in detecting early signs of mental health issues such as depression, anxiety, or stress, thereby enabling timely intervention.
- **Neuromarketing**: By analyzing customer emotions, businesses can refine their marketing strategies, products, and services to better align with consumer preferences and enhance user experiences.
- **Education**: In remote learning environments, FER systems can monitor student engagement, helping educators tailor their teaching methods to improve learning outcomes.
- **Human-Computer Interaction**: Adaptive systems can leverage emotion recognition to provide personalized user experiences, such as adjusting tone in virtual assistants or tailoring content recommendations.

The Role of Computer Vision Algorithms

Computer vision algorithms offer a scalable and efficient solution to the problem of emotion recognition. They can process and classify facial expressions in real time, eliminating the need for manual annotation or intervention. This project leverages deep learning techniques to address key challenges in FER, such as extracting meaningful features from low-resolution images, handling class imbalance, and mitigating the effects of noisy data. By integrating advanced architectures and preprocessing techniques, the project aims to create a lightweight, accurate, and computationally efficient FER system suitable for practical applications.

Through the use of convolutional neural networks (CNNs), transfer learning, and model fine-tuning, this study bridges the gap between raw facial data and actionable insights, paving the way for impactful applications in fields ranging from healthcare to marketing.

Setup and Tools

The project was implemented using Python and TensorFlow, leveraging pre-trained architectures and GPU acceleration for efficient training. The FER-2013 dataset was partitioned into training and testing sets. Each model was trained with a focus on optimizing accuracy while minimizing overfitting and computational cost. Supporting libraries included NumPy, Matplotlib, and Scikit-learn for preprocessing, visualization, and evaluation, respectively.

EXPLORATORY DATA ANALYSIS (EDA) AND PREPROCESSING

Dataset Overview

Link to Dataset: https://www.kaggle.com/datasets/msambare/fer2013

The FER-2013 dataset, sourced from Kaggle, is a widely used benchmark for facial emotion recognition tasks. It contains a total of 35,887 grayscale images, each of resolution 48x48 pixels, organized into two main subsets: a training set comprising 28,709 images and a test set with 7,178 images. The dataset is categorized into seven distinct emotional classes: *Angry*, *Disgust*, *Fear*, *Happy*, *Neutral*, *Sad*, and *Surprise*. While its widespread adoption highlights its relevance, the dataset poses several challenges that make it difficult to achieve high accuracy. These include its inherently low resolution, which limits the granularity of features available for detecting subtle expressions, and a pronounced class imbalance. For example, the *Disgust* class constitutes less than 2% of the total dataset, making it particularly challenging for models to learn and correctly classify this emotion. Such issues necessitate robust preprocessing techniques and architectural innovations to extract meaningful patterns.

Dataset Preprocessing Steps

To address the inherent challenges of the FER-2013 dataset, several preprocessing techniques were employed to improve its quality and suitability for deep learning models. These steps aimed to mitigate the effects of low resolution, imbalance, and noise while enhancing the dataset's variability to support generalization.

1. **Rescaling**: The pixel intensity values of the images were normalized to the range [0, 1] by dividing them by 255. This normalization step standardizes the input data, reducing the risk of numerical instability during training and accelerating convergence by ensuring uniform gradients across layers.

2. **Data Augmentation**: Given the limited resolution and inherent variability in the dataset, data augmentation techniques were applied to artificially expand the diversity of the training data. These included:
    - **Rotation**: Images were randomly rotated by up to 30 degrees to simulate variations in facial orientation.
    - **Horizontal Flipping**: Symmetric flipping was used to reflect faces along the vertical axis, simulating left and right profile perspectives.
    - **Zooming and Cropping**: Random zoom and cropping operations were applied to simulate variations in the distance and framing of facial captures.
    - **Brightness Adjustments**: Variations in brightness were introduced to account for differences in lighting conditions. These augmentations not only improved the model's ability to generalize but also reduced overfitting by preventing the network from memorizing specific features unique to the training set.
3. **Class Weights**: The class imbalance in the dataset was addressed by computing inverse class frequencies to assign higher weights to underrepresented classes during training. For example, the *Disgust* class, with its disproportionately small representation, was given a significantly higher weight compared to the *Happy* class, which had a larger number of samples. These class weights were passed as a parameter to the training function, ensuring that the model paid adequate attention to minority classes during optimization.

By employing these preprocessing techniques, the dataset was effectively transformed into a more robust foundation for training deep learning models. These steps not only enhanced the dataset's quality but also addressed its inherent limitations, facilitating the extraction of meaningful patterns and improving the models' ability to generalize across diverse facial expressions.

MODELING METHODS OVERVIEW

This project implemented and evaluated three distinct models, each contributing to the overall objective of robust emotion recognition. A detailed analysis of each follows an overview of the models, covering their architectures, performance metrics, challenges, and contributions.

Model 1: ResNet50

ResNet50, a deep convolutional neural network pre-trained on ImageNet, was adapted for FER-2013 by incorporating additional layers, including GlobalAveragePooling2D, a dense layer with 1024 neurons, and dropout. Trained for 20 epochs with a batch size of 32, the model provided a strong baseline. However, its high complexity led to long training times and limited generalization.

Model 2: Attention-Based Model with CBAM Blocks

This model integrated Convolutional Block Attention Modules (CBAM) to prioritize essential spatial and channel-wise features dynamically. While the attention mechanism enhanced feature extraction, it introduced significant computational overhead. Trained for 30 epochs with a learning rate scheduler, the model showed only marginal accuracy improvements, underscoring the dataset's limitations.

Model 3: Fine-Tuned MobileNetV2

MobileNetV2, known for its lightweight architecture and efficient depth-wise separable convolutions, was fine-tuned by unfreezing base layers and retraining with a learning rate of 1e-5. Advanced data augmentation and class weights further enhanced performance. This model achieved the highest accuracy, demonstrating its superior adaptability to the FER-2013 dataset.

MODEL 1: RESNET50 MODEL

ResNet50, a widely recognized convolutional neural network architecture, was the first model explored for the emotion recognition task in this project. ResNet50's deep residual learning framework makes it highly effective for extracting features from images, even when applied to datasets with inherent challenges like FER-2013. This model served as a baseline, offering insights into how deeper architectures perform on emotion recognition tasks. Despite its robust design, the model encountered significant difficulties due to the limitations of the FER-2013 dataset and the problem of overfitting.

Model Architecture

ResNet50 is a 50-layer deep convolutional network built on the concept of residual learning. Its architecture includes:

1. **Residual Blocks**: ResNet50 addresses the vanishing gradient problem through shortcut connections that enable residual learning. These connections allow gradients to flow through the network without vanishing, thereby supporting deeper architectures.
    - Each residual block consists of multiple convolutional layers followed by a skip connection that adds the input features to the output.
2. **Feature Extraction Layers**: The convolutional layers capture low- and high-level features such as edges, textures, and patterns across multiple scales.
3. **Global Average Pooling (GAP)**: After feature extraction, GAP reduces the spatial dimensions to a single feature vector while preserving the learned information, enabling compatibility with fully connected layers.
4. **Fully Connected Layers**: The dense layers abstract the features learned by the convolutional layers and output probabilities for each emotion class via a softmax layer with 7 neurons (one for each emotion category).

In this project, the top layer of the pretrained ResNet50 was removed to accommodate custom layers for emotion recognition. These custom layers included a Global Average Pooling layer, a dense layer with 1024 neurons and ReLU activation, and a dropout layer to prevent overfitting. The final output layer used the softmax activation function for multi-class classification.

Preprocessing and Training Setup

The FER-2013 dataset, consisting of 48×48 grayscale facial images, underwent a series of preprocessing steps to prepare it for use with the ResNet50 architecture. Data augmentation was employed to address the limited dataset size and improve the model's ability to generalize to unseen data. Various transformations were applied during training, including random rotations, translations, and zooms, which simulated real-world variations in facial expressions. Horizontal flipping and shearing were also incorporated to ensure the model's robustness to diverse facial orientations and perspectives.

Normalization of pixel values to the range [0, 1] was another critical preprocessing step, as it ensured numerical stability during training and prevented potential issues with gradient updates. To meet the input size requirements of ResNet50, all images were resized to 224×224 pixels, enabling seamless integration with the model's architecture. Finally, the dataset was partitioned into training and validation sets to facilitate effective model training and evaluation. The test set was left unshuffled to ensure consistent and unbiased evaluation of the model's performance. These preprocessing steps were essential in optimizing the dataset for use with a deep learning model as complex as ResNet50.The model was initialized with pretrained weights from ImageNet, leveraging transfer learning to capture rich feature representations. During training, the base ResNet50 layers were frozen, and only the custom top layers were trained. This approach allowed the model to focus on the specific task of emotion recognition without overfitting to the FER-2013 dataset.

Training and Performance

The model was compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy as the evaluation metric. Key hyperparameters included:

- **Learning Rate**: A default rate of 1×10−31 \times 10^{-3} was used to enable efficient weight updates during backpropagation.
- **Batch Size**: Set to 32, balancing memory constraints and gradient estimation.
- **Epochs**: The training ran for 20 epochs, with early stopping to prevent overfitting.

Despite its powerful architecture, the model struggled to achieve high performance due to the FER-2013 dataset's limitations:

- **Test Accuracy**: 24.81%
- **Test Loss**: 1.8012

These results highlight the difficulty of achieving high accuracy on FER-2013 with a computationally intensive model like ResNet50. Figure 1 highlights the **Model Accuracy** and **Model Loss** trends throughout the training and validation phases for this model.

The implementation of ResNet50 for emotion recognition faced several challenges and limitations. One major challenge was overfitting due to the depth and complexity of the architecture. Despite incorporating regularization techniques such as dropout layers and early stopping, the model showed signs of overfitting to the small dataset, hindering its generalizability. Another issue was class imbalance, where underrepresented emotions like "Disgust" and "Fear" were consistently misclassified. This challenge stemmed from the model's inability to effectively learn the distinguishing features of these minority classes. Additionally, the low resolution of the FER-2013 dataset, with images sized at 48×48 pixels, limited the model’s capacity to capture fine-grained details, particularly for subtle or overlapping emotions. Training time also posed a significant

limitation, with each epoch requiring approximately six minutes on a GPU, leading to prolonged training sessions that affected the efficiency of experimentation.

Despite these challenges, ResNet50 demonstrated notable strengths. The use of transfer learning by leveraging ImageNet-pretrained weights allowed the model to begin with robust feature representations, significantly reducing the time required for convergence. Moreover, the model effectively extracted general facial features, making it a strong baseline for downstream tasks when paired with higher-quality datasets. However, ResNet50's high complexity proved to be a disadvantage for the FER-2013 dataset, which lacks the diversity and detail necessary for deep architectures. Furthermore, freezing the base layers during training restricted the model's ability to adapt to the dataset's unique characteristics, resulting in suboptimal performance. These findings underscore the importance of tailoring model complexity to the dataset and problem at hand.

Model 1 Conclusion and Future Improvements

ResNet50 served as a baseline model, setting a benchmark for evaluating subsequent architectures. While it demonstrated the potential of deep learning for emotion recognition, its limitations highlighted the need for alternative approaches. Fine-tuning the ResNet50 layers by unfreezing them could have allowed the model to adapt better to the FER-2013 dataset, improving generalization and accuracy. This strategy would enable pretrained layers to learn dataset-specific features, enhancing overall performance.

Improving input data quality was another critical consideration. Preprocessing techniques such as face alignment and noise reduction could mitigate the effects of low-resolution images, providing cleaner and more informative inputs for the model. Addressing class imbalance was equally important. Strategies like oversampling underrepresented classes, generating synthetic data, or employing specialized loss functions such as focal loss could have ensured more balanced performance across all emotional categories.

Transitioning to lightweight models like MobileNetV2 presented a compelling alternative, offering a balance between computational efficiency and accuracy. These architectures enable faster training while maintaining competitive results, making them a practical choice for the challenges posed by FER-2013.

The insights gained from ResNet50 informed the decision to explore alternative architectures, leading to improved results with the Attention-Based Model and Fine-Tuned MobileNetV2. These models addressed key challenges more effectively, pushing the boundaries of emotion recognition systems.

MODEL 2: ATTENTION-BASED MODEL WITH CBM BLOCK

The Attention-Based Model with Contextual Block Mechanisms (CBM) represents a modern approach to improving feature extraction in convolutional neural networks. This model was specifically designed to address some of the limitations encountered in traditional architectures when dealing with nuanced datasets like FER-2013. By incorporating attention mechanisms, the model dynamically focuses on the most critical parts of an image, enabling better recognition of subtle emotions. This approach aims to mitigate challenges such as overlapping features between emotions and improve the model's ability to generalize across diverse facial expressions.

Model Architecture

The Attention-Based Model integrates a feature extractor backbone with attention blocks to prioritize relevant regions of the input image. The architecture consists of:

1. **Convolutional Layers**: These layers extract low-level features such as edges, textures, and simple patterns from the input image.
2. **Contextual Attention Blocks (CBM)**: This is the core of the model, designed to enhance its ability to identify spatially significant areas within the input image. The attention mechanism assigns higher weights to regions with distinctive features (e.g., eyebrows for "Angry," mouth shape for "Happy").
    - **Channel Attention**: Enhances feature representation by focusing on important channels in the feature map.
    - **Spatial Attention**: Highlights the spatial regions of the feature map that are most relevant to emotion recognition.
3. **Global Average Pooling**: Averages the spatial dimensions of the feature maps to reduce dimensionality and extract robust global features.

4. **Dense Layers**: Includes a fully connected layer with ReLU activation for abstraction, followed by a softmax layer with 7 neurons (one for each emotion category).

This architecture allows the model to attend to relevant parts of the face while suppressing less important regions, making it particularly effective for tasks involving fine-grained classification.

Preprocessing and Training Setup

The FER-2013 dataset was preprocessed with a similar strategy as other models, including normalization and data augmentation. Augmentations such as rotations, translations, zoom, and brightness adjustments were applied to simulate variations in real-world scenarios. The model input size was set to 224×224 pixels, which is standard for most state-of-the-art architectures.

The Attention-Based Model was trained using categorical cross-entropy loss and the Adam optimizer with a learning rate of 1×10−51 \times 10^{-5}. A learning rate scheduler was employed to adjust the learning rate dynamically, allowing the model to converge more effectively over 30 epochs. The training process was monitored using callbacks, including early stopping and model checkpointing, to prevent overfitting and retain the best-performing model.

Performance Metrics

The performance of the Attention-Based Model was moderate compared to other tested models:

- **Test Accuracy**: 26.41%
- **Test Loss**: 1.7746

The model exhibited gradual improvement in training and validation accuracy over the epochs but failed to achieve satisfactory generalization. The attention mechanism performed well in distinguishing dominant emotions like "Happy" and "Surprise," but it struggled with underrepresented and overlapping classes such as

"Disgust" and "Fear." Figure 2 highlights the Model Accuracy **and** Model Loss trends throughout the training and validation phases.

The implementation of the attention-based model with CBM blocks presented both notable challenges and distinct strengths. Among the primary challenges was the high computational cost associated with the integration of attention mechanisms and CBM blocks. This design choice significantly increased the computational requirements, resulting in longer training times. Each epoch required approximately six minutes to complete, which rendered the model less efficient compared to lightweight architectures such as MobileNetV2. Additionally, the issue of class imbalance persisted despite the attention mechanism's capacity to prioritize important features, leading to reduced accuracy for underrepresented emotional classes. Another limitation was the difficulty in distinguishing overlapping features between similar emotions, such as "Sad" and "Neutral." While the attention mechanism aimed to isolate distinguishing characteristics, the low resolution of the FER-2013 dataset constrained its effectiveness in capturing subtle differences.

Despite these challenges, the attention-based model demonstrated important strengths. The incorporation of attention blocks enhanced the model's ability to focus on critical features within facial regions, such as the eyes, mouth, and eyebrows, which are key for emotion recognition. Furthermore, the modular design of the CBM blocks allowed for seamless integration with various architectures, enabling adaptability for future experimentation and potential applications.

However, the model's weaknesses were significant. The relatively low test accuracy of 26.41% highlighted the limitations of the attention mechanism in overcoming the dataset's inherent challenges. Furthermore, its computational inefficiency posed practical limitations, particularly in scenarios where rapid training and inference are essential. These factors collectively emphasized the need for further refinement and exploration of alternative strategies to enhance both accuracy and efficiency in future iterations.

Model 2 Conclusion and Future Improvements

The Attention-Based Model demonstrated the potential of attention mechanisms to enhance feature extraction for emotion recognition. However, the model's performance on FER-2013 was underwhelming compared to simpler architectures. This underperformance can be attributed to the dataset's limitations, such as low resolution and class imbalance.

To further enhance the performance of the emotion recognition models, several advanced strategies can be implemented. First, transitioning to higher-quality datasets such as AffectNet or RAF-DB would provide more detailed, diverse, and balanced data, addressing the limitations posed by the FER-2013 dataset's low resolution and class imbalance. Second, optimizing attention mechanisms by incorporating advanced modules such as multi-head self-attention, as utilized in transformer architectures, could improve the model's ability to capture subtle and nuanced features critical for distinguishing similar emotions. Third, exploring hybrid architectures that combine attention mechanisms with lightweight models like MobileNetV2 could strike a better balance between accuracy and computational efficiency, ensuring practicality for real-world applications. Lastly, incorporating advanced data preprocessing techniques, such as face alignment for better facial region localization and noise removal to clean mislabeled or distorted data, could significantly improve input quality, resulting in enhanced model performance. These strategies collectively represent a pathway to advancing the capabilities of emotion recognition systems.

The Attention-Based Model serves as an exploration into the potential of attention mechanisms for emotion recognition. While it fell short of expectations on FER-2013, its innovative design provides a foundation for future research and development.

MODEL 3: FINE-TUNED MOBILENETV2 MODEL

The Fine-Tuned MobileNetV2 model was selected for its computational efficiency and capability to adapt to resource-constrained environments while maintaining competitive performance. This model leverages the pre-trained MobileNetV2 architecture, initially trained on ImageNet, and fine-tuned it for the emotion recognition task using the FER-2013 dataset. This decision was driven by the need for a lightweight model that balances accuracy and computational efficiency, especially given the challenges posed by the FER-2013 dataset, such as low-resolution images and class imbalance.

Model Architecture

MobileNetV2 is a lightweight convolutional neural network optimized for mobile and embedded devices. It uses depth wise separable convolutions and inverted residuals to reduce computational complexity without compromising feature extraction. For this project, the base MobileNetV2 model was modified by adding custom layers tailored to the FER-2013 dataset. Specifically:

1. **Global Average Pooling**: This layer replaced the fully connected layers to reduce the risk of overfitting and minimize parameters.
2. **Dropout Layers**: Two dropout layers with a rate of 0.5 were added to regularize the model and prevent overfitting.
3. **Dense Layers**: A dense layer with 128 neurons and ReLU activation was included for feature abstraction, followed by the output layer with 7 neurons (one for each emotion class) and a softmax activation function.

The base MobileNetV2 model was fully unfrozen to allow fine-tuning. This enabled the model to adjust its pre-trained weights based on the FER-2013 dataset, extracting features that are more relevant to emotion recognition.

Preprocessing and Training Setup

The dataset was augmented using a range of transformations, including random rotations, zoom, shear,

and horizontal flips. This augmentation aimed to enhance the model's generalization ability by simulating real-world variations in facial expressions. Pixel values were normalized to the [0, 1] range to ensure compatibility with the pre-trained model.

The model was trained using a categorical cross-entropy loss function and the Adam optimizer with a learning rate of 1×10−51 \times 10^{-5}. A learning rate scheduler was implemented to decrease the learning rate exponentially after 10 epochs, ensuring gradual convergence. Training was conducted for 30 epochs, and early stopping was used to prevent overfitting by monitoring validation loss with a patience of 5 epochs.

Performance Metrics

The Fine-Tuned MobileNetV2 achieved the highest performance among the models tested:

- **Test Accuracy**: 50.54%
- **Test Loss**: 1.5230

The learning curve (Figure 3) exhibited consistent improvement in both training and validation accuracy, indicating effective learning without overfitting. The confusion matrix (Figure 4) revealed strong performance for classes like "Happy" and "Surprise" but challenges in distinguishing between "Fear" and "Disgust," reflecting the dataset's inherent difficulties.

Challenges and Solutions

The model encountered several challenges during its development and evaluation. First, class imbalance posed a significant issue, particularly with underrepresented emotions such as "Disgust." This imbalance was addressed by incorporating class weights during training to ensure these minority classes were adequately emphasized. Second, the low resolution of the dataset, consisting of 48×48-pixel grayscale images, limited the model's ability to discern fine-grained facial features. Fine-tuning the model allowed it to adapt and perform effectively within these constraints. Lastly, overlapping visual features among certain emotions, such as "Sad" and "Neutral," often led to misclassifications. To mitigate this, an extensive data augmentation strategy was

employed, exposing the model to a diverse range of variations to improve its robustness and generalization capabilities.

Model 3 Conclusion and Future Improvements

The Fine-Tuned MobileNetV2 model demonstrated its adaptability and efficiency, outperforming both the ResNet50 and Attention-Based models in terms of accuracy and computational cost. Future improvements could include using higher-resolution datasets like AffectNet or RAF-DB, which provide more detailed facial features. Additionally, implementing ensemble techniques or experimenting with modern architectures like Vision Transformers could further enhance performance. The model's success underscores the importance of fine-tuning lightweight architectures for domain-specific tasks.

COMPARATIVE ANALYSIS OF ALL THE MODELS

A comprehensive comparative analysis of the three models is provided in Table 1, summarizing their performance across key metrics such as accuracy, loss, architectural complexity, and training time. Each model underwent rigorous evaluation to identify strengths, limitations, and overall suitability for facial emotion recognition tasks. MobileNetV2, the fine-tuned lightweight model, demonstrated superior performance, balancing computational efficiency and classification accuracy, with a significantly improved test accuracy of 50.54%.

Table 1. Model Comparison

Feature/Metric	Model 1: ResNet50	Model 2: Attention-Based	Model 3: Fine-Tuned MobileNetV2
Test Accuracy (%)	24.81	26.41	50.54
Validation Accuracy (%)	24.71	25.41	42.67
Test Loss	1.8012	1.7746	1.5230
Validation Loss	1.8068	1.7846	1.3410
Architectural Complexity	High	High	Moderate
Training Time/Epoch	5–6 minutes	>6 minutes	~2.5 minutes

The iterative development process from Model 1 to Model 3 underscores the critical importance of refining architectures and techniques to enhance performance. ResNet50 (Model 1) established a robust baseline, leveraging a powerful pretrained architecture; however, its complexity and high training time per epoch limited its adaptability to the FER-2013 dataset's challenges, such as low resolution and class imbalance. The inability to

sufficiently generalize to the test set reflected in its low accuracy and high loss. Attention-Based (Model 2) introduced feature prioritization with an attention mechanism and CBM blocks, which aimed to enhance the model's ability to focus on emotionally salient regions in facial images. Despite its theoretical strengths, this model faced substantial computational overhead, and the marginal improvement in accuracy (26.41%) did not justify its complexity or training time, which exceeded six minutes per epoch. These results indicated that the model struggled to fully leverage the dataset's limited resolution and variability. Fine-Tuned MobileNetV2 (Model 3) emerged as the most effective solution, capitalizing on a lightweight architecture combined with fine-tuning. By unfreezing layers and employing a lower learning rate, the model adapted effectively to the dataset-specific challenges, achieving significant gains in both accuracy and loss metrics. Its computational efficiency, requiring approximately 2.5 minutes per epoch, further underscored its practicality for deployment in resource-constrained environments.

Figure 5 provides a graphical comparison of the training and validation accuracy and loss trends for all three models, offering an in-depth analysis of their convergence behaviors. The ResNet50 model showed stagnation in validation metrics, highlighting overfitting tendencies. The Attention-Based Model exhibited modest improvements in generalization but faced slower convergence. In contrast, MobileNetV2 demonstrated steady improvements, achieving a better balance between training and validation metrics.

This comparative analysis highlights the progression of improvements made across models. ResNet50's complexity, while powerful, proved inefficient for the task. The Attention-Based Model introduced an innovative architectural enhancement but suffered from computational inefficiency and limited performance gains. Fine-Tuned MobileNetV2, with its lightweight design and effective optimization strategies, showcased the ideal balance for addressing the challenges of emotion recognition in the FER-2013 dataset. Future directions include exploring ensemble learning to combine the complementary strengths of these architectures and experimenting with higher-quality datasets to overcome the inherent limitations of FER-2013.

PROJECT CHALLENGES AND FUTURE IMPROVEMENTS

Challenges

Several challenges were encountered throughout this project. The FER-2013 dataset posed significant limitations due to its low resolution (48×48 pixels), class imbalance, and noisy labels. These factors made it difficult for models to distinguish subtle differences between similar emotions, such as "Sad" and "Neutral." The high computational cost of ResNet50 and the Attention-Based Model further complicated the training process. While MobileNetV2 mitigated some of these issues, achieving higher accuracy remained challenging due to the dataset's inherent constraints.

Future Improvements

To enhance the performance of emotion recognition systems, several advanced strategies can be employed, each targeting specific aspects of the challenges faced in model training and evaluation. These strategies focus on improving the dataset, exploring cutting-edge architectures, leveraging ensemble techniques, employing robust augmentation strategies, and refining the fine-tuning process.

Enhancing Dataset Quality

Transitioning to higher-quality datasets like AffectNet or RAF-DB can significantly improve model performance by addressing some of the inherent limitations of the FER-2013 dataset. AffectNet, for instance, provides a more extensive collection of facial expression images with higher resolution, better labeling accuracy, and a wider range of emotional categories. Similarly, RAF-DB offers realistic facial expressions captured under diverse conditions, which enhances the generalization ability of models trained on such datasets. High-quality datasets reduce noise and ambiguity, allowing models to learn more discriminative features for subtle emotions.

Advancing Model Architectures

Exploring advanced neural network architectures, such as Vision Transformers (ViTs) or hybrid models combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, can push the boundaries of emotion recognition. Vision Transformers excel at capturing long-range dependencies within image data, enabling precise feature extraction. Hybrid models like CNN-LSTM architectures incorporate both spatial and temporal information, making them particularly effective for dynamic emotion recognition tasks where sequential data is involved. Such models can better capture subtle variations in facial expressions that static images might miss.

Leveraging Ensemble Learning

Ensemble learning offers an effective way to combine the strengths of multiple models. By aggregating predictions from architectures like ResNet50 and MobileNetV2, ensemble techniques can mitigate individual model weaknesses and enhance overall accuracy. For instance, ResNet50's ability to extract deep hierarchical features can complement MobileNetV2's lightweight efficiency. Weighted averaging or majority voting mechanisms can be applied to optimize the ensemble's predictions, yielding a more robust and reliable emotion recognition system.

Advanced Augmentation Techniques

Sophisticated data augmentation strategies can further improve model robustness. Techniques like occlusion-based augmentations, where parts of the face are intentionally masked, can prepare the model to handle real-world scenarios involving partially visible faces. Additionally, adversarial training, which introduces perturbations to input data, can enhance the model's ability to resist noise and adversarial attacks. These techniques ensure that the model generalizes well across varying lighting conditions, angles, and facial occlusions.

Refining Fine-Tuning Processes

Fine-tuning the model with specialized loss functions can address challenges like class imbalance more effectively. For example, focal loss focuses on hard-to-classify samples, emphasizing underrepresented classes like *Disgust* or *Fear*. This approach ensures that the model does not disproportionately prioritize majority classes. Additionally, experimenting with varying learning rates or adaptive optimizers like AdamW can further refine the model's weight updates during training, enhancing convergence and performance.

Incorporating these strategies will not only improve emotion recognition accuracy but also extend the applicability of the system to more challenging and realistic scenarios. By addressing both data and architectural limitations, these enhancements pave the way for the development of robust, efficient, and scalable emotion recognition systems suitable for diverse applications.

CONCLUSION

This study demonstrated the iterative process of refining emotion recognition models, highlighting the complexities and challenges associated with the FER-2013 dataset. Through the evaluation of three distinct architectures—ResNet50, Attention-Based Model, and Fine-Tuned MobileNetV2—insights were gained into the trade-offs between computational complexity, accuracy, and robustness. Among these, the Fine-Tuned MobileNetV2 emerged as the most effective architecture, achieving a balanced performance that optimized accuracy while remaining computationally efficient. This was achieved by leveraging transfer learning, fine-tuning the pretrained model, and employing enhanced data augmentation and class balancing techniques, all of which contributed to addressing the inherent challenges of the FER-2013 dataset, such as class imbalance and low resolution.

Despite its success, the performance of MobileNetV2 underscores the limitations imposed by the FER-2013 dataset, particularly its low image quality, class imbalance, and inherent label noise. Future research should prioritize the use of higher-quality datasets such as AffectNet or RAF-DB, which offer larger sample sizes, more diverse emotional expressions, and higher resolution. These datasets would enable the development of models capable of capturing more nuanced emotional features and achieving significantly higher accuracy. Additionally, the exploration of advanced architectures, such as Vision Transformers (ViTs) and ConvNext, or hybrid approaches combining Convolutional Neural Networks (CNNs) with Long Short-Term Memory Networks (LSTMs), could further enhance the ability to model temporal and spatial dependencies in facial expressions.

To push the boundaries of emotion recognition further, future work should also investigate ensemble learning strategies, which combine the strengths of multiple models to improve generalizability and robustness. Moreover, integrating domain-specific techniques such as facial alignment, occlusion handling, and noise reduction can refine input data quality, enabling more accurate predictions. Hyperparameter optimization using automated tools like Bayesian optimization or grid search could also uncover optimal configurations for specific datasets, ensuring maximum performance.

The potential applications of emotion recognition technology are vast and transformative, spanning domains such as neuromarketing, mental health assessment, and human-computer interaction. In neuromarketing, emotion recognition can revolutionize consumer behavior analysis by providing real-time insights into emotional responses to advertisements and products. In mental health, such models could assist clinicians in identifying subtle emotional cues indicative of disorders like depression or anxiety, offering a non-invasive and scalable diagnostic tool. In human-computer interaction, emotion-aware systems can enhance user experiences by adapting responses based on emotional states, fostering more natural and intuitive interactions.

By addressing the limitations of existing datasets and architectures and embracing modern advancements in machine learning and computer vision, this field holds immense promise. The iterative approach demonstrated in this study provides a robust foundation for future innovations, paving the way for more reliable, efficient, and impactful emotion recognition systems.

REFERENCES

Articles and Books

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
- Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). AffectNet: A database for facial expression, valence, and arousal computing in the wild. IEEE Transactions on Affective Computing, 10(1), 18–31. https://doi.org/10.1109/TAFFC.2017.2740923
- Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.
- Wang, W., Liang, X., Chen, Z., & Lin, L. (2020). Attention-based convolutional neural networks for emotion recognition in video. IEEE Transactions on Multimedia, 22(2), 515–526. https://doi.org/10.1109/TMM.2019.2933525

Datasets and Tools

- FER-2013 Dataset. (2013). Kaggle. Retrieved from https://www.kaggle.com/datasets/deadskull7/fer2013
- AffectNet. (2017). Affective Computing Dataset. Retrieved from http://www.affectnet.org

One Person Group A Final Report Appendix

AAI-521 Applied Computer Vision, University of San Diego

APPENDIX

Link to Github Repo: http://www.github.com/arifakokab/Deep-Vision-Emotions-Recognition-

Figure 1: Model Accuracy and Loss Trend for ResNet50 Model:

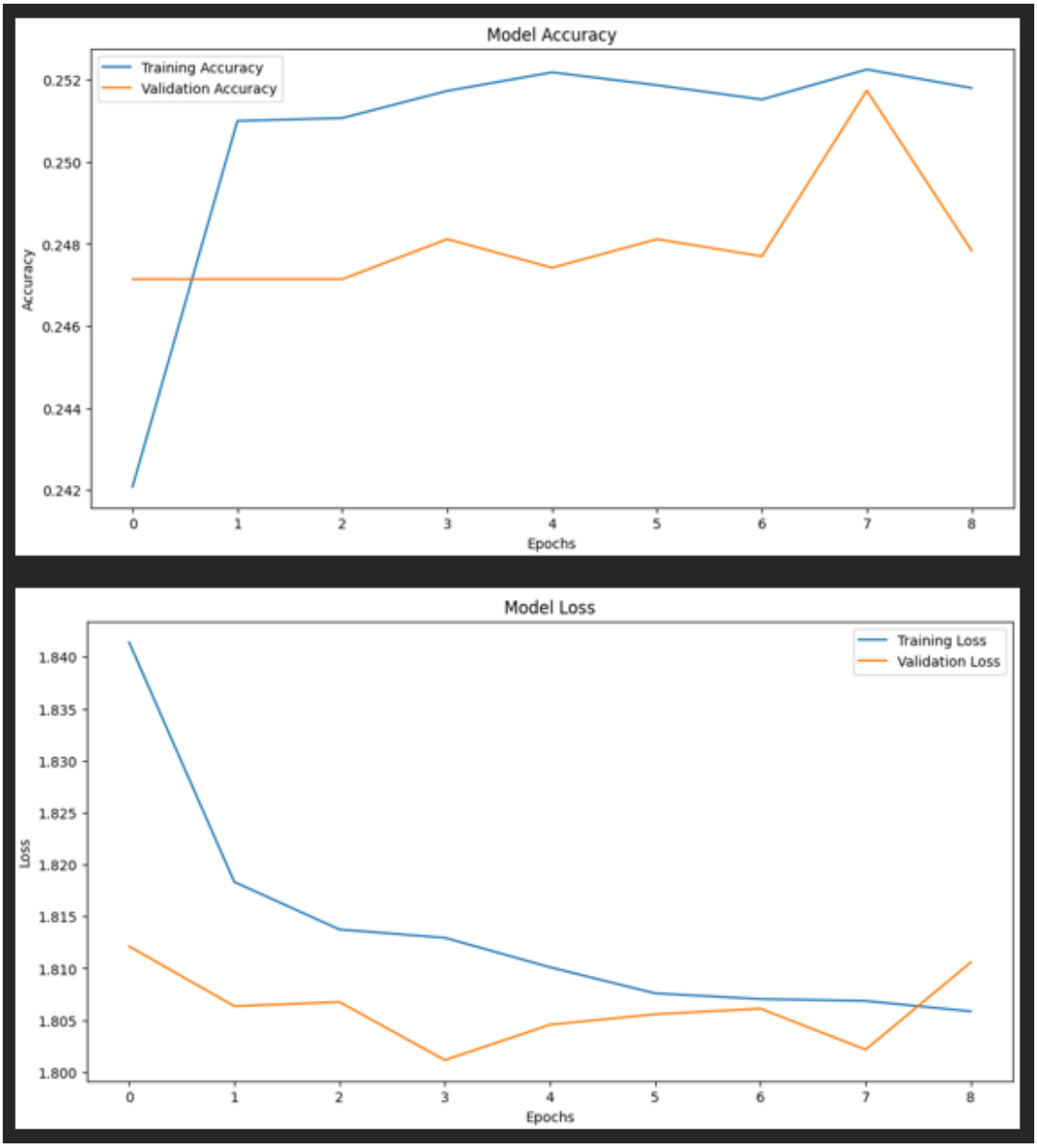


Figure 2: Model Accuracy and Loss Trend for Attention Based Model with CBM Block:

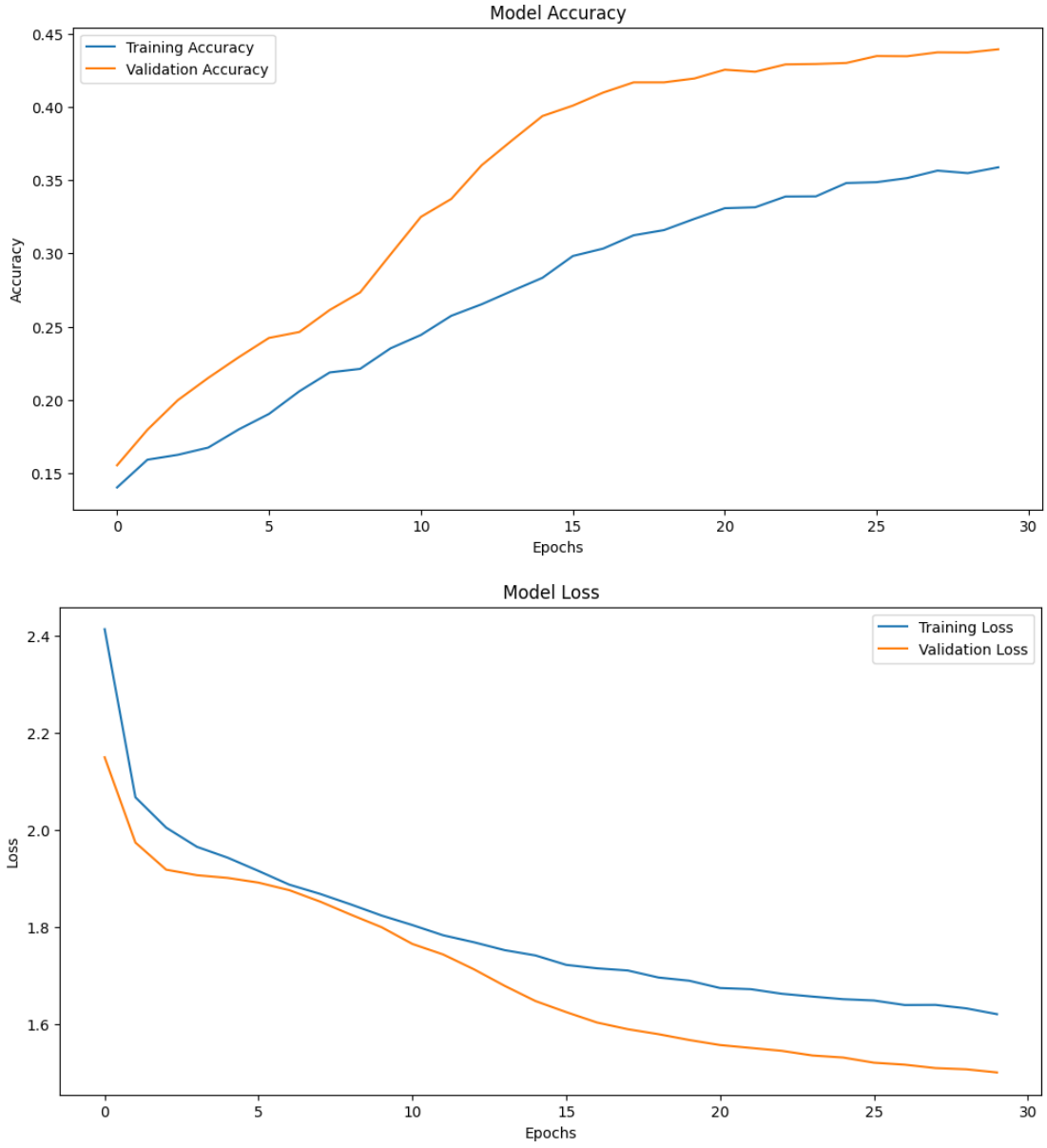


Figure 3: Model Accuracy and Loss Trend for Fine-Tune MobileNetV2 Model:

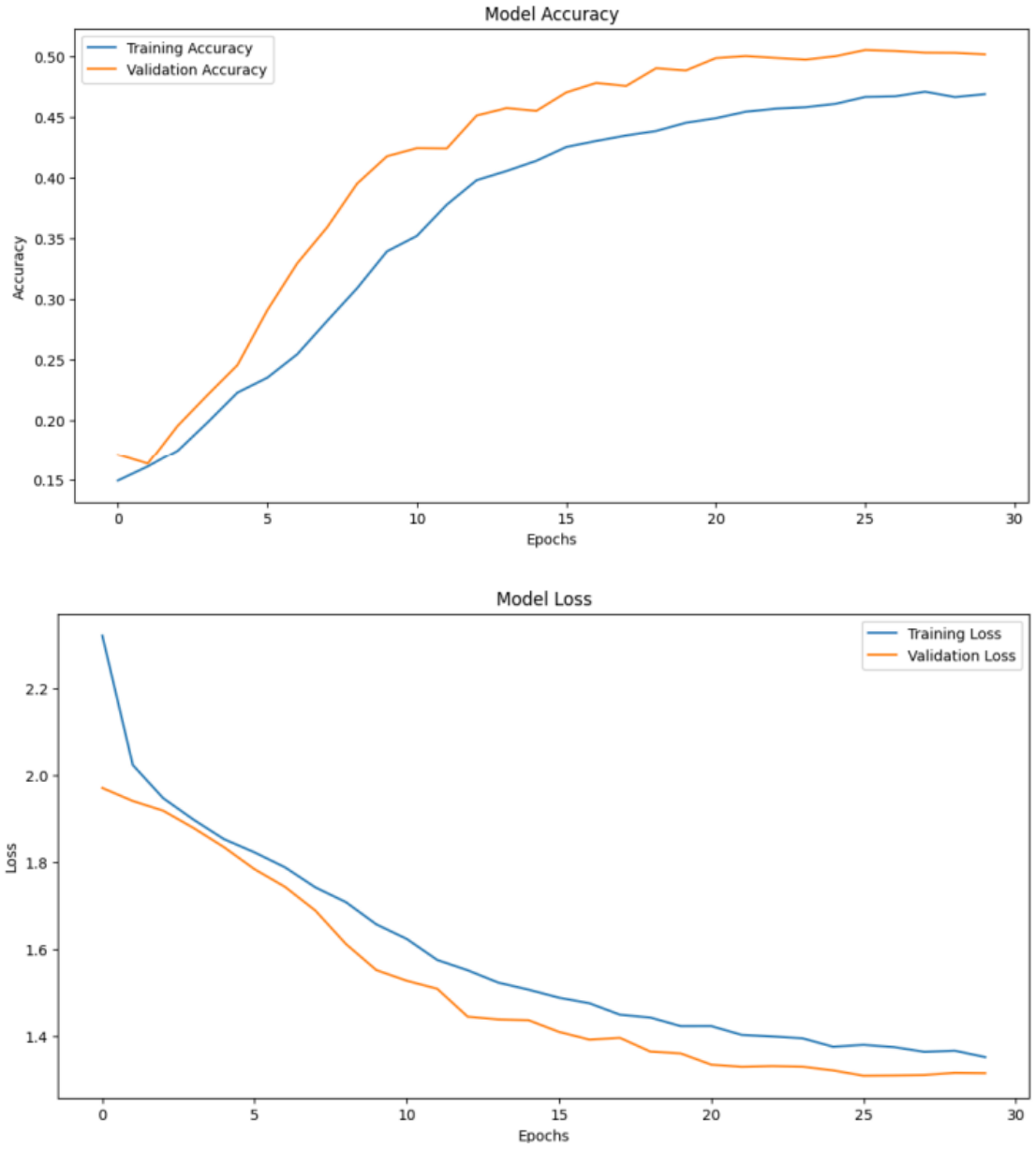


Figure 4 Confusion Matrix for Fine-Tuned MobileNetV2 Model:

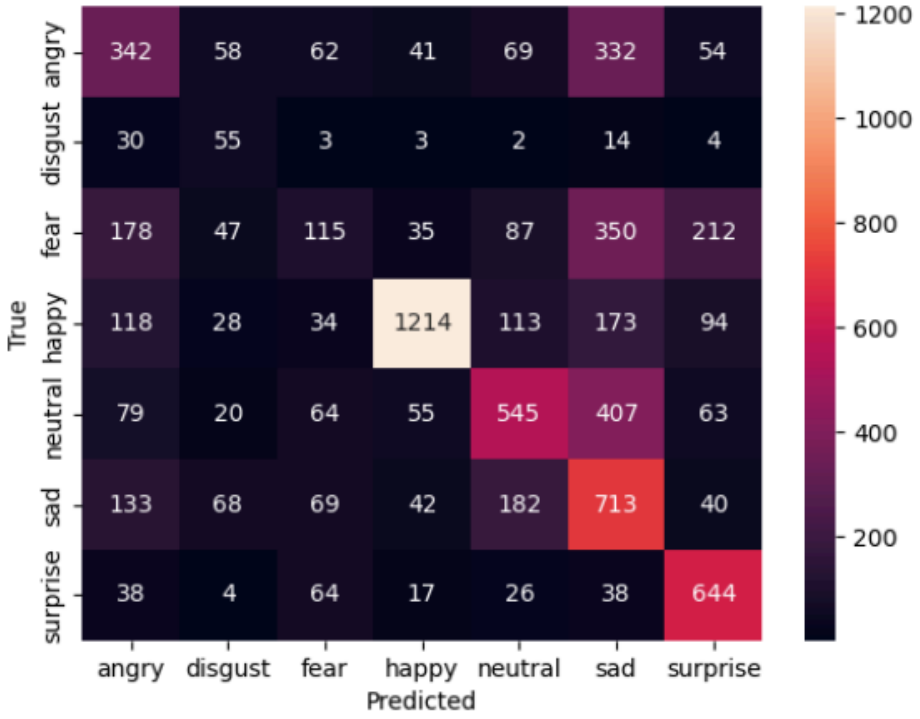
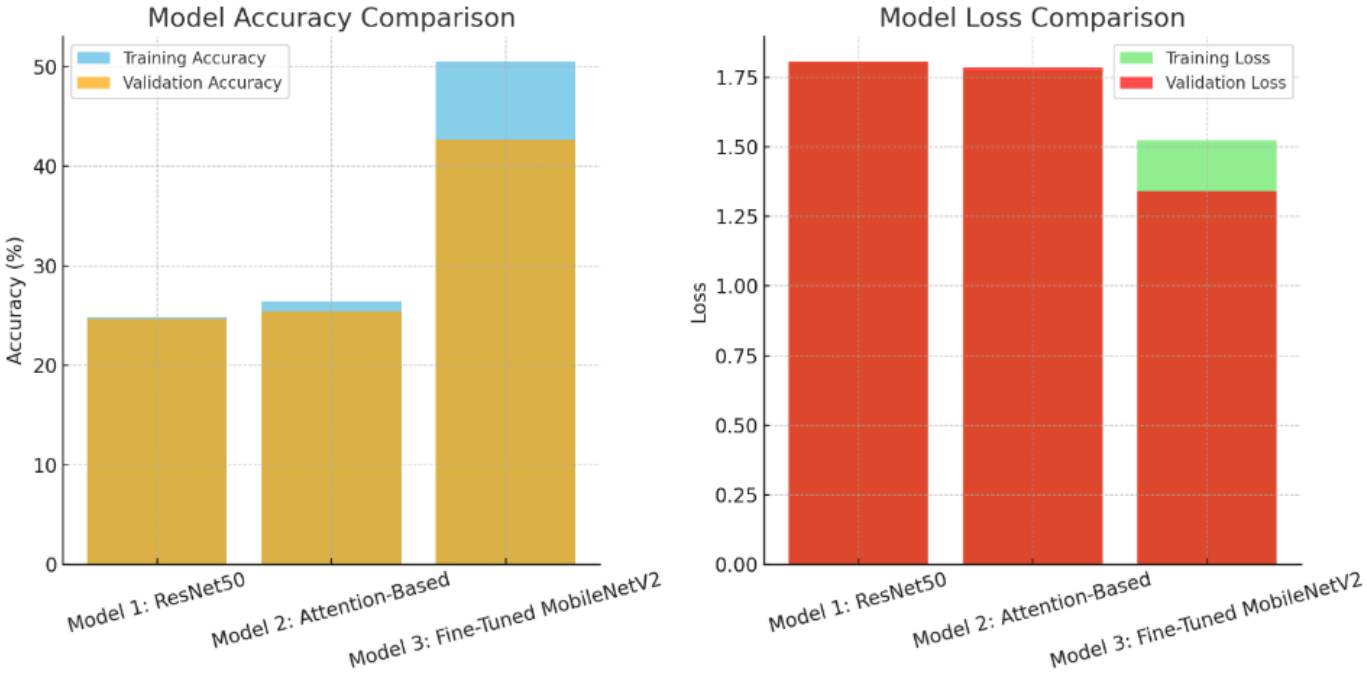


Figure 5 Model Accuracy and Loss Comparison of all Three Models:



Arifa Kokab - Final Project AAI-521

```
from google.colab import files
files.upload()  # Upload your kaggle.json file
```

```
Choose Files | kaggle.json
• kaggle.json(application/json) - 70 bytes, last modified: 12/3/2024 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"arifakokab6939","key":"1605554cf0e8a918ca2d83985c976e6d"}'}
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d msambare/fer2013
!unzip fer2013.zip -d /content/dataset
```

```
  inflating: /content/dataset/train/surprise/Training_98446939.jpg
  inflating: /content/dataset/train/surprise/Training_98447563.jpg
  inflating: /content/dataset/train/surprise/Training_98472640.jpg
  inflating: /content/dataset/train/surprise/Training_98481914.jpg
  inflating: /content/dataset/train/surprise/Training_98511054.jpg
  inflating: /content/dataset/train/surprise/Training_98514521.jpg
  inflating: /content/dataset/train/surprise/Training_98539906.jpg
  inflating: /content/dataset/train/surprise/Training_98569897.jpg
  inflating: /content/dataset/train/surprise/Training_98576352.jpg
  inflating: /content/dataset/train/surprise/Training_98588726.jpg
  inflating: /content/dataset/train/surprise/Training_98594534.jpg
  inflating: /content/dataset/train/surprise/Training_98644924.jpg
  inflating: /content/dataset/train/surprise/Training_98657632.jpg
  inflating: /content/dataset/train/surprise/Training_98664620.jpg
  inflating: /content/dataset/train/surprise/Training_98737557.jpg
  inflating: /content/dataset/train/surprise/Training_98747340.jpg
  inflating: /content/dataset/train/surprise/Training_98750930.jpg
  inflating: /content/dataset/train/surprise/Training_98758885.jpg
  inflating: /content/dataset/train/surprise/Training_98767792.jpg
  inflating: /content/dataset/train/surprise/Training_98774734.jpg
  inflating: /content/dataset/train/surprise/Training_9880069.jpg
  inflating: /content/dataset/train/surprise/Training_98827007.jpg
  inflating: /content/dataset/train/surprise/Training_98828949.jpg
  inflating: /content/dataset/train/surprise/Training_98840937.jpg
  inflating: /content/dataset/train/surprise/Training_98861159.jpg
  inflating: /content/dataset/train/surprise/Training_98899707.jpg
  inflating: /content/dataset/train/surprise/Training_9890261.jpg
  inflating: /content/dataset/train/surprise/Training_98971238.jpg
  inflating: /content/dataset/train/surprise/Training_98972491.jpg
  inflating: /content/dataset/train/surprise/Training_99066709.jpg
  inflating: /content/dataset/train/surprise/Training_9909756.jpg
  inflating: /content/dataset/train/surprise/Training_99104204.jpg
  inflating: /content/dataset/train/surprise/Training_99106305.jpg
  inflating: /content/dataset/train/surprise/Training_99113691.jpg
  inflating: /content/dataset/train/surprise/Training_991508.jpg
  inflating: /content/dataset/train/surprise/Training_99182527.jpg
  inflating: /content/dataset/train/surprise/Training_99233432.jpg
  inflating: /content/dataset/train/surprise/Training_99267647.jpg
  inflating: /content/dataset/train/surprise/Training_99279802.jpg
  inflating: /content/dataset/train/surprise/Training_99318660.jpg
  inflating: /content/dataset/train/surprise/Training_9932811.jpg
  inflating: /content/dataset/train/surprise/Training_99399304.jpg
  inflating: /content/dataset/train/surprise/Training_99399823.jpg
  inflating: /content/dataset/train/surprise/Training_99404349.jpg
  inflating: /content/dataset/train/surprise/Training_9943398.jpg
  inflating: /content/dataset/train/surprise/Training_99441101.jpg
  inflating: /content/dataset/train/surprise/Training_99576616.jpg
  inflating: /content/dataset/train/surprise/Training_99604118.jpg
  inflating: /content/dataset/train/surprise/Training_99663019.jpg
  inflating: /content/dataset/train/surprise/Training_99791966.jpg
  inflating: /content/dataset/train/surprise/Training_99794165.jpg
  inflating: /content/dataset/train/surprise/Training_99807705.jpg
  inflating: /content/dataset/train/surprise/Training_99827442.jpg
  inflating: /content/dataset/train/surprise/Training_99916297.jpg
  inflating: /content/dataset/train/surprise/Training_99924420.jpg
  inflating: /content/dataset/train/surprise/Training_99937001.jpg
  inflating: /content/dataset/train/surprise/Training_99951755.jpg
  inflating: /content/dataset/train/surprise/Training_99984132.jpg
```

```
!ls /content/dataset
```

```
test  train
```

```
!ls /content/dataset/train
!ls /content/dataset/test
```

```
angry  disgust  fear  happy  neutral  sad  surprise
angry  disgust  fear  happy  neutral  sad  surprise
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateScheduler
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from tensorflow.keras.applications import MobileNetV2
from sklearn.utils.class_weight import compute_class_weight
```

```
# Hyperparameters
IMG_SIZE = 224
BATCH_SIZE = 32
NUM_CLASSES = 7
EPOCHS = 30
LEARNING_RATE = 1e-5

# Paths to training and test data
train_path = "/content/dataset/train"
test_path = "/content/dataset/test"

# Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

test_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode="categorical"
)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    shuffle=False
)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

```
# Load Pretrained MobileNetV2 and fine-tune
base_model = tf.keras.applications.MobileNetV2(
    weights="imagenet",
    include_top=False,
    input_shape=(IMG_SIZE, IMG_SIZE, 3)
)

# Unfreeze all layers for fine-tuning
base_model.trainable = True

# Add custom layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)  # Add dropout for regularization
x = Dense(128, activation="relu")(x)
```

```
x = Dropout(0.5)(x)
output = Dense(NUM_CLASSES, activation="softmax")(x)

model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

# Compute class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)

class_weights = dict(enumerate(class_weights))
print("Class Weights:", class_weights)

# Define learning rate scheduler
def scheduler(epoch, lr):
    if epoch < 10:
        return float(lr)
    else:
        return float(lr * tf.math.exp(-0.1))

# Define callbacks
lr_callback = tf.keras.callbacks.LearningRateScheduler(scheduler)
callbacks = [
    lr_callback,
    tf.keras.callbacks.ModelCheckpoint("best_model.keras", monitor="val_accuracy", save_best_only=True, mode="max"),
    tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, restore_best_weights=True)
]
```

```
Class Weights: {0: 1.0266046844269623, 1: 9.406618610747051, 2: 1.0010460615781582, 3: 0.5684387684387684, 4: 0.8260394187886635, 5: 0.8491274770777877, 6: 1.293372978330405}
```

```
# Train the Model
history = model.fit(
    train_generator,
    validation_data=test_generator,
    epochs=30,
    callbacks=callbacks,
    class_weight=class_weights
)
```

```
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 336ms/step - accuracy: 0.1612 - loss: 2.0322 - val_accuracy: 0.1636 - val_loss: 1.9399 - learning_rate: 1.0000e-05
Epoch 3/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 338ms/step - accuracy: 0.1686 - loss: 1.9727 - val_accuracy: 0.1956 - val_loss: 1.9175 - learning_rate: 1.0000e-05
Epoch 4/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 313s 346ms/step - accuracy: 0.1895 - loss: 1.9140 - val_accuracy: 0.2211 - val_loss: 1.8777 - learning_rate: 1.0000e-05
Epoch 5/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 312s 345ms/step - accuracy: 0.2169 - loss: 1.8564 - val_accuracy: 0.2458 - val_loss: 1.8336 - learning_rate: 1.0000e-05
Epoch 6/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 302s 334ms/step - accuracy: 0.2300 - loss: 1.8306 - val_accuracy: 0.2910 - val_loss: 1.7831 - learning_rate: 1.0000e-05
Epoch 7/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 336ms/step - accuracy: 0.2467 - loss: 1.8022 - val_accuracy: 0.3296 - val_loss: 1.7426 - learning_rate: 1.0000e-05
Epoch 8/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 306s 338ms/step - accuracy: 0.2768 - loss: 1.7425 - val_accuracy: 0.3593 - val_loss: 1.6879 - learning_rate: 1.0000e-05
Epoch 9/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 316s 349ms/step - accuracy: 0.3024 - loss: 1.7142 - val_accuracy: 0.3952 - val_loss: 1.6114 - learning_rate: 1.0000e-05
Epoch 10/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 316s 349ms/step - accuracy: 0.3322 - loss: 1.6682 - val_accuracy: 0.4178 - val_loss: 1.5513 - learning_rate: 1.0000e-05
Epoch 11/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 307s 339ms/step - accuracy: 0.3458 - loss: 1.6136 - val_accuracy: 0.4245 - val_loss: 1.5268 - learning_rate: 9.0484e-06
Epoch 12/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 306s 339ms/step - accuracy: 0.3766 - loss: 1.5902 - val_accuracy: 0.4242 - val_loss: 1.5084 - learning_rate: 8.1873e-06
Epoch 13/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 337ms/step - accuracy: 0.3948 - loss: 1.5484 - val_accuracy: 0.4514 - val_loss: 1.4443 - learning_rate: 7.4082e-06
Epoch 14/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 307s 339ms/step - accuracy: 0.4003 - loss: 1.5402 - val_accuracy: 0.4575 - val_loss: 1.4379 - learning_rate: 6.7032e-06
Epoch 15/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 312s 345ms/step - accuracy: 0.4081 - loss: 1.5233 - val_accuracy: 0.4553 - val_loss: 1.4361 - learning_rate: 6.0653e-06
Epoch 16/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 313s 346ms/step - accuracy: 0.4303 - loss: 1.4849 - val_accuracy: 0.4705 - val_loss: 1.4093 - learning_rate: 5.4881e-06
Epoch 17/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 302s 334ms/step - accuracy: 0.4358 - loss: 1.4819 - val_accuracy: 0.4783 - val_loss: 1.3916 - learning_rate: 4.9659e-06
Epoch 18/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 303s 335ms/step - accuracy: 0.4363 - loss: 1.4476 - val_accuracy: 0.4758 - val_loss: 1.3957 - learning_rate: 4.4933e-06
Epoch 19/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 302s 334ms/step - accuracy: 0.4370 - loss: 1.4423 - val_accuracy: 0.4904 - val_loss: 1.3641 - learning_rate: 4.0657e-06
Epoch 20/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 313s 346ms/step - accuracy: 0.4407 - loss: 1.4291 - val_accuracy: 0.4886 - val_loss: 1.3598 - learning_rate: 3.6788e-06
Epoch 21/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 318s 351ms/step - accuracy: 0.4557 - loss: 1.4229 - val_accuracy: 0.4987 - val_loss: 1.3340 - learning_rate: 3.3287e-06
Epoch 22/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 329s 363ms/step - accuracy: 0.4564 - loss: 1.3930 - val_accuracy: 0.5004 - val_loss: 1.3291 - learning_rate: 3.0119e-06
Epoch 23/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 307s 339ms/step - accuracy: 0.4588 - loss: 1.3669 - val_accuracy: 0.4989 - val_loss: 1.3307 - learning_rate: 2.7253e-06
Epoch 24/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 337ms/step - accuracy: 0.4565 - loss: 1.3919 - val_accuracy: 0.4975 - val_loss: 1.3294 - learning_rate: 2.4660e-06
Epoch 25/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 306s 338ms/step - accuracy: 0.4602 - loss: 1.3689 - val_accuracy: 0.5003 - val_loss: 1.3209 - learning_rate: 2.2313e-06
Epoch 26/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 314s 347ms/step - accuracy: 0.4640 - loss: 1.3807 - val_accuracy: 0.5054 - val_loss: 1.3086 - learning_rate: 2.0190e-06
Epoch 27/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 313s 346ms/step - accuracy: 0.4653 - loss: 1.3796 - val_accuracy: 0.5046 - val_loss: 1.3092 - learning_rate: 1.8268e-06
Epoch 28/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 302s 334ms/step - accuracy: 0.4716 - loss: 1.3528 - val_accuracy: 0.5032 - val_loss: 1.3103 - learning_rate: 1.6530e-06
Epoch 29/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 305s 337ms/step - accuracy: 0.4631 - loss: 1.3757 - val_accuracy: 0.5031 - val_loss: 1.3152 - learning_rate: 1.4957e-06
Epoch 30/30
898/898 ━━━━━━━━━━━━━━━━━━━━ 302s 334ms/step - accuracy: 0.4713 - loss: 1.3409 - val_accuracy: 0.5018 - val_loss: 1.3144 - learning_rate: 1.3534e-06
```

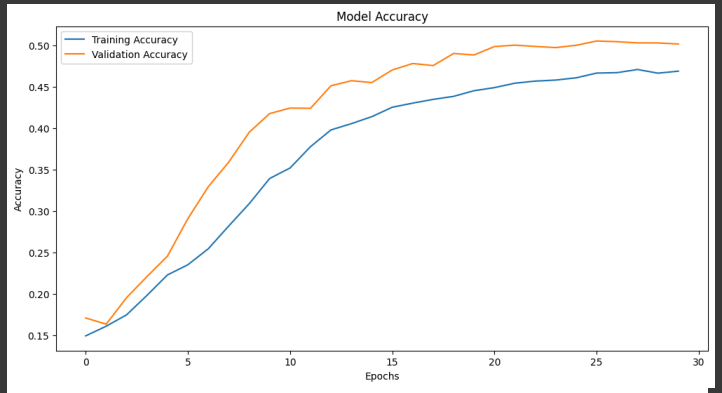
```
# Save the final model
model.save("mobilenetv2_fine_tuned_final.keras")
```

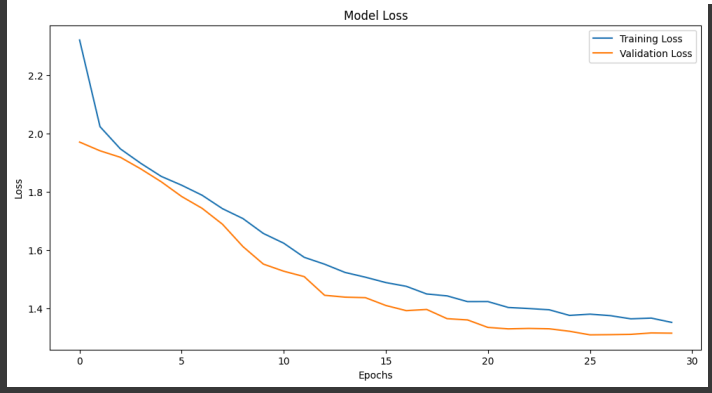
```
# Evaluate the Model
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 7s 29ms/step - accuracy: 0.4030 - loss: 1.4141
Test Loss: 1.3085689544677734
Test Accuracy: 0.5054332613945007
```

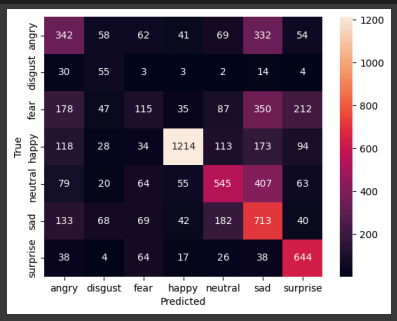
```
# Visualize Training History
plt.figure(figsize=(12, 6))
plt.plot(history.history["accuracy"], label="Training Accuracy")
plt.plot(history.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Model Accuracy")
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("Model Loss")
plt.show()
```






```
#Generate Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d', xticklabels=test_generator.class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```




```
#Delve deeper into model performance

# Generate predictions on the test set
y_pred = np.argmax(model.predict(test_generator), axis=1)  # Predicted class indices
y_true = test_generator.classes  # True class labels

# Map class indices to emotion labels
class_labels = list(test_generator.class_indices.keys())  # Get class names

# Classification report
report = classification_report(y_true, y_pred, target_names=class_labels, digits=4)
print("Classification Report:")
print(report)

# Confusion Matrix
print("Confusion Matrix:")
cm = confusion_matrix(y_true, y_pred)
print(cm)
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 6s 28ms/step
Classification Report:
              precision    recall  f1-score   support

       angry     0.3725    0.3570    0.3646       958
     disgust     0.1964    0.4955    0.2813       111
        fear     0.2798    0.1123    0.1603      1024
       happy     0.8628    0.6843    0.7633      1774
     neutral     0.5322    0.4420    0.4829      1233
         sad     0.3518    0.5718    0.4356      1247
    surprise     0.5797    0.7750    0.6632       831

    accuracy                         0.5054      7178
   macro avg     0.4536    0.4911    0.4502      7178
weighted avg     0.5256    0.5054    0.4999      7178

Confusion Matrix:
[[ 342   58   62   41   69  332   54]
 [  30   55    3    3    2   14    4]
 [ 178   47  115   35   87  350  212]
 [ 118   28   34 1214  113  173   94]
 [  79   20   64   55  545  407   63]
 [ 133   68   69   42  182  713   40]
 [  38    4   64   17   26   38  644]]
```