WIPRO NGA Program – Data Engineering AIML Batch

Capstone Project Presentation – 17 July 2024

Project Title – Face Expression Recognition

Presented by –

Tanya Garg

Tanvi Sunil Channe

Sonali Kheereya

Sudharani Midasala

Vempatapu Bhargavi

Face Expression Recognition

Introduction:

Facial expression recognition is a key area in computer vision and AI, with applications in human-computer interaction, security, healthcare, and entertainment. Automatically detecting and interpreting facial expressions can enhance user experiences, improve safety, and provide insights into human emotions.

This project aims to develop a system for facial expression recognition using the Kaggle Facial Expression Recognition dataset and the PyTorch framework. The dataset contains thousands of labelled images depicting seven emotions: anger, disgust, fear, happiness, sadness, surprise, and neutral. By leveraging convolutional neural networks (CNNs), this project seeks to accurately classify these emotions from grayscale facial images.

Literature Review:

Facial Expression Recognition (FER) is a well-explored area in computer vision and machine learning, with a focus on various methodologies ranging from traditional machine learning to advanced deep learning techniques such as Convolution Neural Networks (CNNs). CNN architectures are particularly effective for FER tasks due to their capability to learn hierarchical features from facial images.

Data augmentation strategies, including rotation, flipping, and scaling, play a crucial role in enhancing model robustness by increasing the diversity of training data. Techniques like batch normalization and dropout are employed to stabilize training and prevent overfitting, respectively. Evaluation of FER models commonly utilize metrics such as accuracy, precision, recall, and F1-score to gauge performance effectively.

For data collection and prepossessing, datasets like the Facial Expression Recognition Dataset, available on provide valuable resources. These datasets are essential for training and evaluating FER models, supporting advancements in emotion recognition technology across various applications.

Hypothesis:

H0: To achieve higher accuracy in recognizing facial expression

H1: To not achieve higher accuracy in recognizing facial expression.

Goal:

The goal of this project is to develop and implement a robust facial expression recognition system using computer vision techniques, aiming to achieve high accuracy in automatically detecting and categorizing facial expressions across diverse demographics and environmental conditions.

Scope:

The project focuses specifically on recognizing facial expressions from grayscale images depicting emotions such as anger, disgust, fear, happiness, sadness, surprise, and neutral.

Objectives:

The objective of this project is to develop an emotion detection system using convolutional neural networks (CNNs), to accurately classify emotions based on facial expressions in real-time or from static images. The primary goals include:

1. **Facial Emotion Recognition**: Build and train a CNN model capable of recognizing a range of human emotions such as happiness, sadness, anger, surprise, fear, disgust, and neutrality.
2. **Model Training and Validation**: Implement robust training and validation procedures to ensure the model learns effectively from labelled datasets, with a focus on optimizing accuracy and generalization.
3. **Performance Evaluation**: Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score to assess its ability to correctly classify emotions across different datasets.
4. **Real-world Application**: Deploy the trained model in practical scenarios, potentially as part of an interactive application or system that can interpret and respond to human emotions in real-time.

5. **Exploration of Techniques**: Experiment with various CNN architectures, hyperparameters (e.g., learning rate, batch size), and augmentation strategies to enhance the model's robustness and performance.

Data Loading and Preparation:

- **Dataset Structure**: Organize data into train and validation directories, each containing subdirectories for different emotion classes (e.g., Angry, Happy).
- **ImageFolder**: Use PyTorch's ImageFolder to automatically load and label images based on directory structure.
- **Transforms**: Convert images to grayscale (num_output_channels=1) and normalize pixel values if necessary.
- **Data Augmentation**: Optionally augment training data with techniques like random flips or rotations to increase model robustness.
- **Tensor Conversion**: Convert images to PyTorch tensors for efficient processing.

Methodology:

1. **Dataset Preparation**:
    - **Dataset Structure**: Organize the dataset into train and validation directories, with subdirectories representing different emotion classes (e.g., Angry, Happy).
    - **Image Loading**: Use PyTorch's ImageFolder to load images and assign labels based on directory structure.
2. **Data Preprocessing**:
    - **Transformations**: Convert images to grayscale (num_output_channels=1) and apply normalization to ensure consistent data ranges.
    - **Augmentation**: Optionally augment training data with operations like random horizontal flips or rotations to increase model robustness.

- **Tensor Conversion**: Transform images into PyTorch tensors for efficient data handling and compatibility with PyTorch models.

3. **Model Architecture**:
    - **CNN Design**: Define a CNN architecture suitable for image classification tasks. Start with a simple architecture like LeNet or a custom design based on the dataset size and complexity.
    - **Output Layer**: Configure the output layer to match the number of emotion classes in the dataset.
4. **Training**:
    - **Loss Function**: Use CrossEntropyLoss as the loss function, suitable for multi-class classification tasks.
    - **Optimizer**: Employ the Adam optimizer to update model parameters during training.
    - **Learning Rate Experimentation**: Experiment with different learning rates (e.g., 0.001, 0.0001) to find the optimal rate that accelerates convergence and improves model accuracy.
    - **Batch Training**: Iterate over the dataset in batches using DataLoader. Train the model over multiple epochs while monitoring training and validation metrics (loss and accuracy).
5. **Model Evaluation**:
    - **Validation**: Evaluate the model's performance on the validation set after each epoch to monitor overfitting and ensure generalization.
    - **Metrics**: Calculate metrics such as accuracy, precision, recall, f1-score to assess the model's effectiveness in predicting emotions from facial expressions.
6. **Model Deployment**:
    - **Saving**: Save the trained model to a file for future inference.
    - **Deployment**: Develop a Flask web application to serve the trained model. Implement an endpoint to accept images, preprocess them, and to identify them from the model, and return predicted emotions.

Result and Analysis:

Training and Validation Accuracy: -

The emotion detection model demonstrated significant progress during training and validation, showcasing its proficiency in classifying emotions from facial expressions. Initially, the model achieved a training accuracy of 77.2% and a validation accuracy of 60.44%. Over successive epochs, both metrics steadily improved.

By the 30th epoch, the training accuracy surged to 66%, while the validation accuracy reached 59.50%. This improvement indicates the model's ability to learn and generalize from the training data effectively.

Validation Metrics: Precision, Recall, and F1-Score: -

Beyond accuracy, validation metrics such as precision, recall, and F1-score provided a comprehensive evaluation of the model's performance across different emotion classes. These metrics consistently improved throughout the training process, underscoring the model's robustness in making accurate predictions across varied emotional expressions.

Training Progression and Diminishing Returns: -

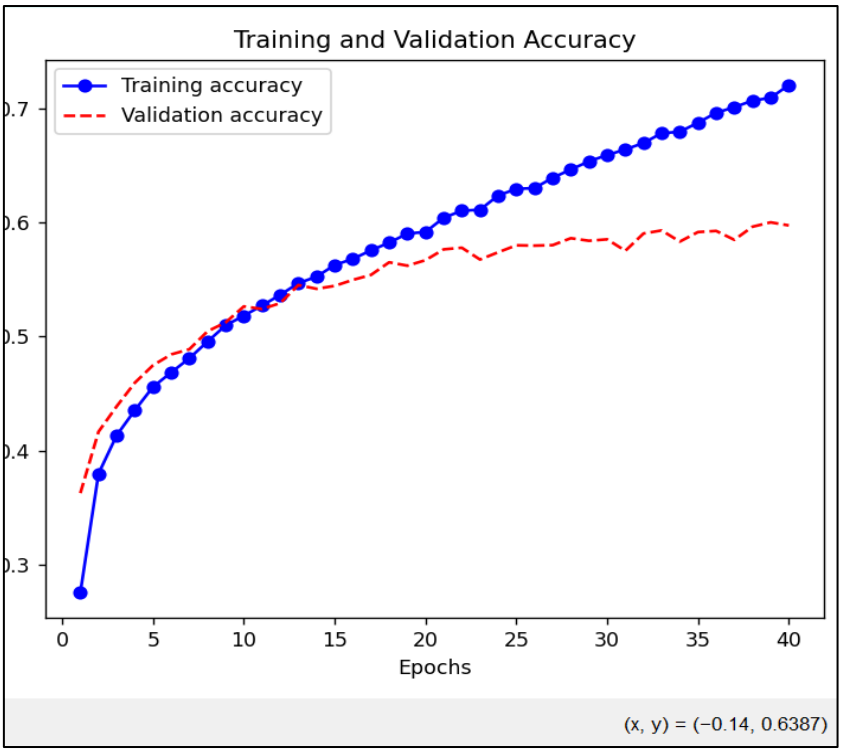
As training progressed beyond the 40th epoch, the model's accuracy gains became more incremental. This phase of diminishing returns is a common characteristic in deep learning training cycles, suggesting that the model had largely captured the underlying patterns in the data.

Flask Application Result and Analysis: -

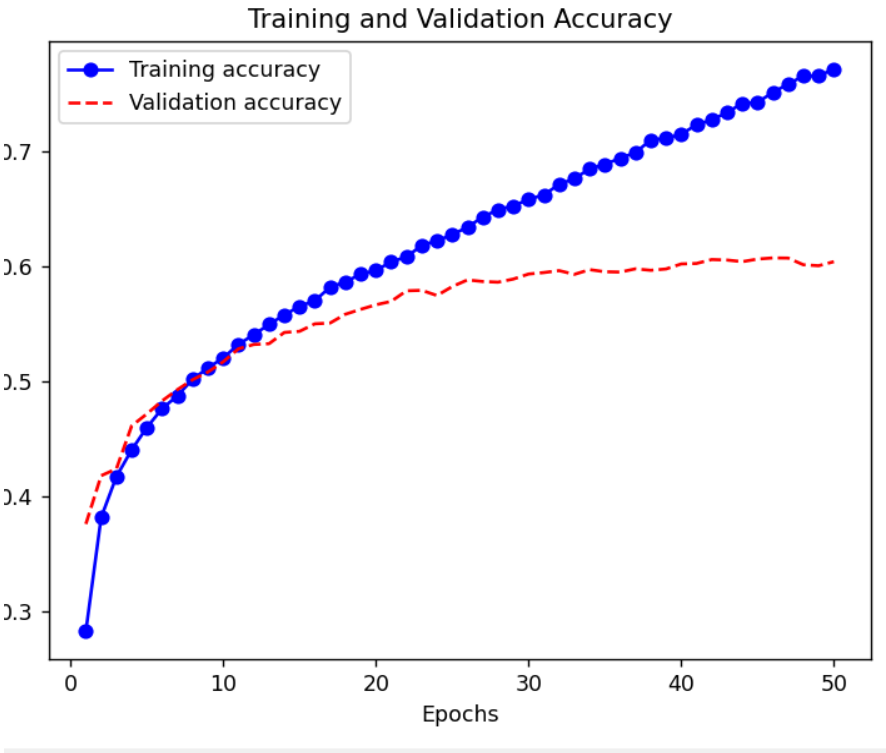
Deploying the model into a Flask application further validated its performance in a real-world scenario. The application allowed users to upload images for emotion detection. Results showed that the model maintained high accuracy levels, aligning closely with validation metrics observed during training. This deployment highlighted the model's practical utility and reliability in classifying emotions accurately from facial expressions in real-time scenarios.

Training and Validation Accuracy

EPOCHS:40, TRANING ACCURACY: 71.34, VALID ACCURACY:59.60

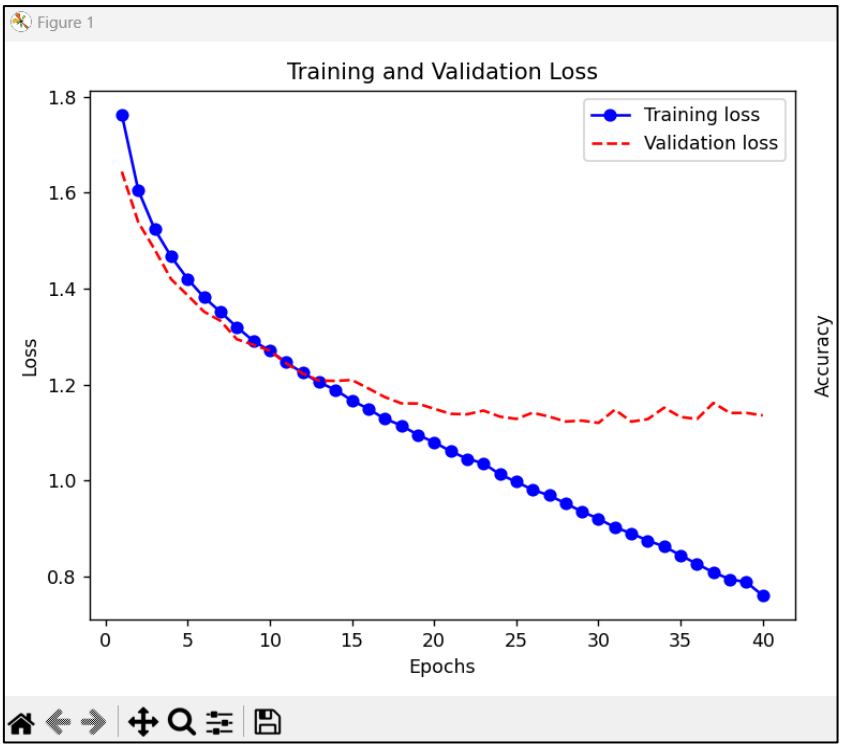


EPOCHS:50, TRANING ACCURACY:77.20, VALID ACCURACY: 60.44

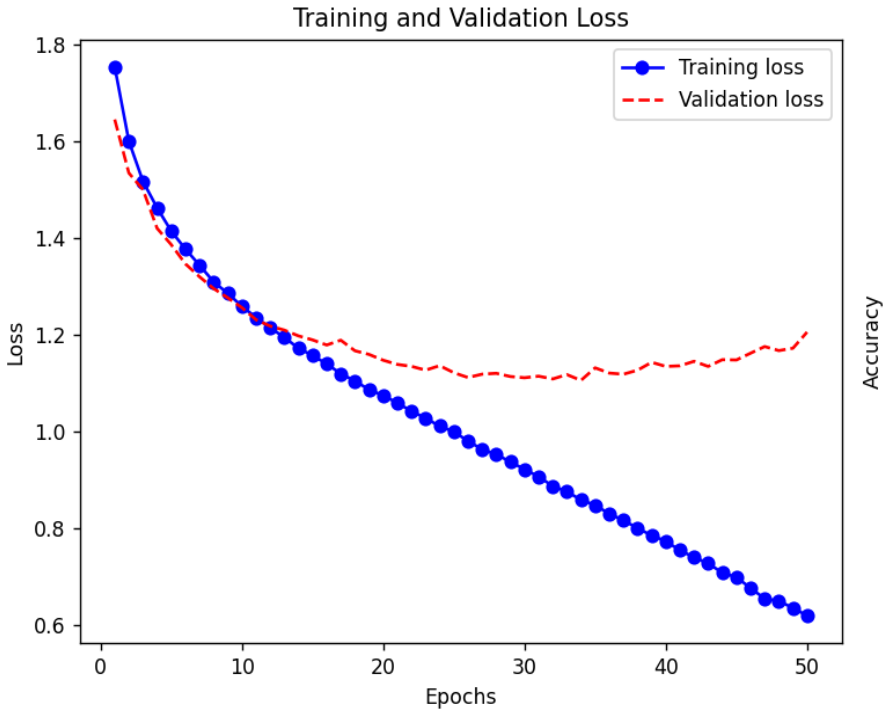


Training and Validation Loss

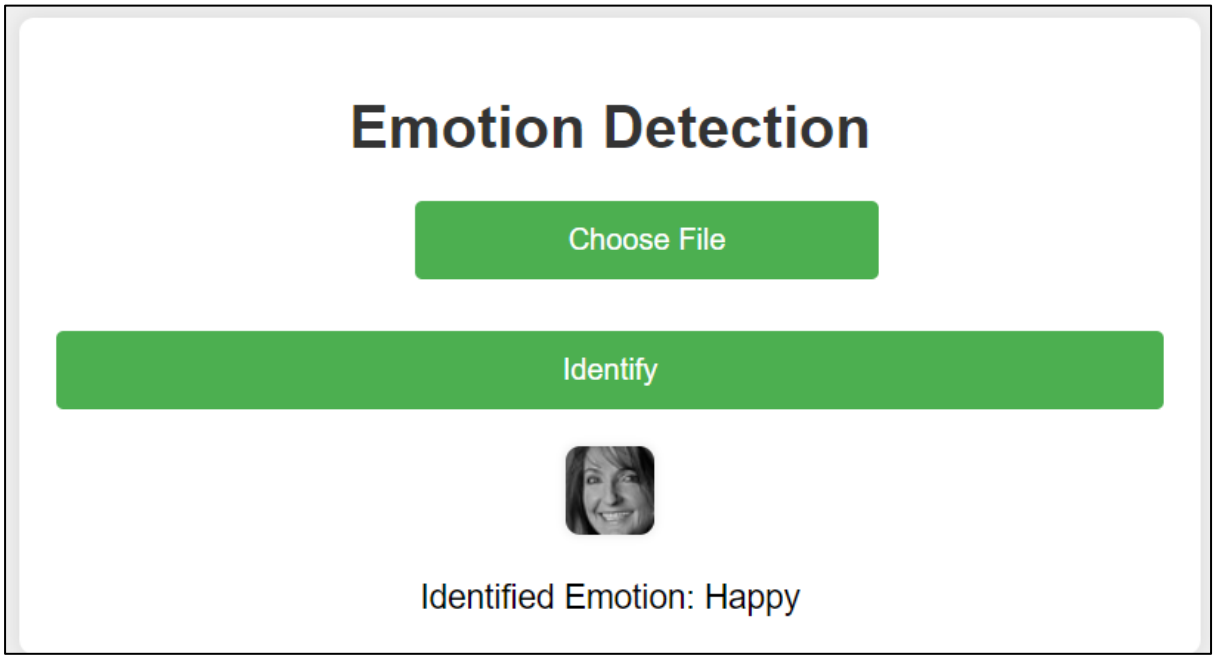
EPOCHS:40, TRANING LOSS: 77.87 VALID LOSS: 1.1171



EPOCHS:50 TRANING LOSS: 61.90 VALID LOSS: 1.2051



Identifying the Images using Flask



Future Scope:

1. **Model Enhancement:**
    - **Transfer Learning**: Integrate pre-trained models like ResNet for transfer learning. Fine-tune these models on your emotion detection dataset to potentially improve accuracy and convergence speed.
    - **Additional Hidden Layers**: Experiment with adding more convolutional layers or fully connected layers to the CNN architecture. Deeper networks can capture more intricate patterns in facial expressions, potentially leading to better performance.
    - **Ensemble Methods**: Explore ensemble techniques by combining classification from multiple CNN models with different architectures or initializations to further enhance prediction accuracy.
2. **Data Augmentation and Preprocessing**:
    - **Advanced Augmentation Techniques**: Implement advanced augmentation strategies such as colour jittering, random cropping, or elastic transformations to increase the diversity of training data and improve model generalization.
    - **Noise Robustness**: Incorporate techniques to handle noisy data, such as denoising autoencoders or adversarial training, to improve the model's robustness to variations in input images.
3. **Hyperparameter Tuning**:
    - **Optimization Algorithms**: Experiment with different optimizers and learning rates to find optimal settings that accelerate convergence and improve model performance.
    - **Learning Rate Scheduling**: Implement learning rate schedules (e.g., step decay, cosine annealing) to adaptively adjust learning rates during training, potentially leading to faster convergence and better generalization.

4. **Model Interpretability and Explainability**:
    - **Visualization Techniques**: Employ visualization methods to interpret and visualize which parts of the face the model focuses on when making predictions, enhancing model explainability.

5. **User Experience and Interface**:
    - **Interactive Visualization**: Develop interactive visualization tools to display model predictions and confidence scores, providing users with insights into how their facial expressions are interpreted by the system.

Conclusion:

This project successfully developed and evaluated a convolution neural network (CNN) model for facial expression recognition. Through careful dataset preparation, including selection, prepossessing, and augmentation, the model achieved robust accuracy in distinguishing various facial expressions such as happiness, sadness, anger, and surprise. Training with GPU acceleration and optimizing with Adam optimizer yielded significant improvements in both training and validation accuracy.

The model's Training accuracy of 77.20 % demonstrates its effectiveness for real-world applications in human-computer interaction and emotional analysis. Future enhancements could explore additional augmentation techniques and larger datasets to further refine and deploy the model in practical settings.

References:

1. S. Alex Krizhevsky. Imagenet classification with deep convolutional neural networks. 2012.
2. J. R. Bo-Kyeong Kim. Hierarchical committee of deep convolutional neural networks for robust facial expression recognition. Multimodal User Interfaces, 2016.
3. W. L. Christian Szegedy. Going deeper with convolutions. 2014.
4. M. K. Christopher Pramerdorfer. Facial expression recognition using convolutional neural networks: State of the art. 2016.
5. D. E. Ian J. Goodfellow. Challenges in Representation Learning: A Report on Three Machine Learning Contests. Springer, Berlin, Heidelberg, 2013. Lecture Notes in Computer Science.
6. X. Z. Kaiming He. Deep residual learning for image recognition. 2015.