Facial Emotion Recognition Using Deep Learning

Project Summary

Name: Tito Varghese

Email : tito.varghese1992@gmail.com

CNN Deep Learning Classifier Model Workflow

1. **Project Overview and Background**
2. **Dataset Overview**
3. **Project Goal**
4. **Data Collection and Preprocessing**
5. **Model Development**
6. **Training and Evaluation**
7. **Real-Time Processing**
8. **Application Development**
9. **Performance Optimization**
10. **Conclusion**
11. **References**

Please paste the GitHub Repo link.

GitHub Profile Link: - https://github.com/7692TITO

GitHub Repository Link:
-https://github.com/VargheseTito/Facial-Emotion-Recognition-App

Facial Emotion Recognition Using Deep Learning

Overview

DeepFER: Facial Emotion Recognition Using Deep Learning aims to develop a robust and efficient system for recognizing emotions from facial expressions using advanced deep learning techniques. This project leverages Convolutional Neural Networks (CNNs) and Transfer Learning to accurately classify emotions such as happiness, sadness, anger, surprise, and more from images of human faces. The system will be trained on a diverse dataset of facial images, employing data augmentation and fine-tuning methods to enhance its performance. By integrating state-of-the-art computer vision algorithms and neural network architectures, DeepFER seeks to achieve high accuracy and real-time processing capabilities. The ultimate goal is to create a versatile tool that can be applied in various fields, including human-computer interaction, mental health monitoring, and customer service, enhancing the way machines understand and respond to human emotions.

Project Background

Customer satisfaction in the e-commerce sector is a pivotal metric that influences loyalty, repeat business, and word-of-mouth marketing. Traditionally, companies have relied on direct surveys to gauge customer satisfaction, which can be time-consuming and may not always capture the full spectrum of customer experiences. With the advent of deep learning, it's now possible to predict customer satisfaction scores in real-time, offering a granular view of service performance and identifying areas for immediate improvement.

In recent years, the field of facial emotion recognition has gained significant attention due to its wide range of applications in various domains, including mental health monitoring, human-computer interaction, customer service, and security. Emotion recognition from facial expressions is a challenging task, as it involves accurately identifying subtle differences in facial features corresponding to different emotional states. Traditional methods relied heavily on handcrafted features and rule-based approaches, which often lacked the ability to generalize across diverse datasets and real-world scenarios.

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized the way facial emotion recognition systems are developed. CNNs have demonstrated exceptional performance in image classification tasks by automatically learning hierarchical feature representations from raw data.

This project, **DeepFER: Facial Emotion Recognition Using Deep Learning**, aims to harness the power of CNNs and Transfer Learning to build a robust and efficient facial emotion recognition system. By training the model on large, annotated datasets and employing advanced techniques such as data augmentation and fine-tuning, DeepFER aspires to achieve high accuracy and real-time processing capabilities.

The motivation behind this project stems from the growing need for automated systems that can understand and respond to human emotions effectively. Such systems can significantly enhance user experiences in various applications, from interactive virtual assistants to personalized mental health interventions. DeepFER seeks to bridge the gap between advanced AI techniques and practical emotion recognition applications, paving the way for more intuitive and empathetic machine interactions with humans.

Dataset Overview

- **Dataset Composition:**
  - Contains images categorized into seven distinct emotion classes: angry, sad, happy, fear, neutral, disgust, and surprise.
- **Emotion Classes:**
  - **Angry:** Images depicting expressions of anger.
  - **Sad:** Images depicting expressions of sadness.

- ➢ **Happy:** Images depicting expressions of happiness.
- ➢ **Fear:** Images depicting expressions of fear.
- ➢ **Neutral:** Images depicting neutral, non-expressive faces.
- ➢ **Disgust:** Images depicting expressions of disgust.
- ➢ **Surprise:** Images depicting expressions of surprise.

- ❖ **Image Characteristics:**
  - ➢ High-quality facial images with diverse backgrounds and lighting conditions.
  - ➢ Includes both posed and spontaneous expressions to ensure robustness.
- ❖ **Data Augmentation:**
  - ➢ Techniques such as rotation, scaling, and flipping applied to increase dataset variability and enhance model generalization.
- ❖ **Dataset Annotations:**
  - ➢ Each image is labeled with its corresponding emotion class.
- ❖ **Data Source:**
  - ➢ Collected from publicly available facial expression databases and crowd-sourced contributions.
- ❖ **Usage:**
  - ➢ Used for training, validation, and testing phases in the emotion recognition model development.
- ❖ **Purpose:**
  - ➢ To train and evaluate the DeepFER model for accurate and real-time facial emotion recognition across diverse scenarios.

Project Goal

The primary goal of DeepFER: Facial Emotion Recognition Using Deep Learning is to develop an advanced and efficient system capable of accurately identifying and classifying human emotions from facial expressions in real-time. By leveraging state-of-the-art Convolutional Neural Networks (CNNs) and Transfer Learning techniques, this project aims to create a robust model that can handle the inherent variability in facial expressions and diverse image conditions. The system will be trained on a comprehensive dataset featuring seven distinct emotions: angry, sad, happy, fear, neutral, disgust, and surprise. The ultimate objective is to achieve high accuracy and reliability, making DeepFER suitable for applications in human-computer interaction, mental health monitoring, customer service, and beyond. Through this project, we aim to bridge the gap between cutting-edge AI research and practical emotion recognition applications, contributing to more empathetic and responsive machine interactions with humans.

Data Collection and Preprocessing

Data Collection

Assemble a Diverse Dataset To build a robust facial emotion recognition model, we began by assembling a diverse dataset that accurately represents the variability in human facial expressions. This dataset includes high-quality images categorized into seven distinct emotions:

- **Angry**: Expressions showing anger, with furrowed brows, tight lips, and glaring eyes.
- **Sad**: Expressions of sadness, often characterized by downturned lips, drooping eyelids, and a lack of brightness in the eyes.
- **Happy**: Expressions of happiness, typically featuring broad smiles, crinkled eyes, and overall brightened facial features.
- **Fear**: Expressions of fear, with wide-open eyes, raised eyebrows, and an open mouth.
- **Neutral**: Non-expressive faces that do not exhibit any specific emotion, providing a baseline for comparison.
- **Disgust**: Expressions of disgust, often shown with a wrinkled nose, raised upper lip, and lowered eyebrows.
- **Surprise**: Expressions of surprise, characterized by wide eyes, raised eyebrows, and an open mouth.

Image Sources The images were sourced from a variety of publicly available facial expression databases, as well as through crowd-sourced contributions to ensure a wide range of demographics and contexts. This approach ensured that our dataset included diverse backgrounds, lighting conditions, and both posed and spontaneous expressions to improve the model's robustness and generalization capabilities.

Data Augmentation

Enhancing Dataset Variability To further enhance the dataset and improve the generalization ability of our model, we implemented several data augmentation techniques. These techniques help the model learn to recognize emotions under various conditions and from different perspectives:

- **Rotation**: Randomly rotating images within a certain range to simulate different head tilts.
- **Scaling**: Scaling images to various sizes to help the model handle different face sizes and distances.
- **Flipping**: Horizontally flipping images to provide mirror images, helping the model recognize emotions from different angles.
- **Brightness and Contrast Adjustment**: Varying the brightness and contrast of images to simulate different lighting conditions.

- **Translation**: Shifting images horizontally or vertically to simulate slight movements of the face within the frame.

These augmentation techniques were applied during the training phase to create a more varied dataset, allowing the model to learn from a wider range of examples and thus improving its ability to generalize to new, unseen data.

Data Preprocessing

Normalization and Resizing

- **Normalization**: All images were normalized to a range of [0, 1] by scaling pixel values. This step ensures that the model trains more efficiently and converges faster by standardizing the input data.
- **Resizing**: Each image was resized to a consistent dimension of 48x48 pixels, which is a standard input size for many facial emotion recognition models. This resizing helps maintain a balance between computational efficiency and the retention of important facial details.

Grayscale Conversion

- Since emotion recognition primarily relies on the structural features of the face rather than color information, all images were converted to grayscale. This reduces computational complexity while preserving essential features necessary for emotion detection.

By carefully collecting a diverse dataset and implementing robust data augmentation and preprocessing techniques, we laid a strong foundation for training our facial emotion recognition model. These steps ensure that the model is well-equipped to handle the variability and complexity inherent in real-world facial expressions, leading to improved accuracy and generalization in practical applications.

Model Development

Custom Convolutional Neural Network (CNN) Architecture

To achieve accurate facial emotion recognition, we designed and implemented a custom Convolutional Neural Network (CNN) architecture tailored specifically for this task. This custom architecture is built using TensorFlow's Keras library and includes several layers designed to capture the intricate features of facial expressions. Here's an overview of the custom CNN model architecture:

1. **Input Layer**:
    - The model starts with an input layer that takes in grayscale images of size 48x48 pixels.
2. **Convolutional Layers**:
    - **First Convolutional Layer**:
        - 32 filters, 3x3 kernel size, ReLU activation.
        - Followed by a Batch Normalization layer to stabilize and speed up the training process.
    - **Second Convolutional Layer**:

        - 64 filters, 3x3 kernel size, ReLU activation.
        - Batch Normalization and a Max Pooling layer (2x2 pool size) to reduce the spatial dimensions.
        - A Dropout layer with a 0.25 dropout rate to prevent overfitting.
    - **Third and Fourth Convolutional Layers**:
        - 128 filters each, 3x3 kernel size, ReLU activation.
        - Each followed by Batch Normalization.
        - Another Max Pooling layer and Dropout (0.25 dropout rate).
    - **Fifth and Sixth Convolutional Layers**:
        - 256 filters each, 3x3 kernel size, ReLU activation.
        - Each followed by Batch Normalization.
        - Another Max Pooling layer and Dropout (0.25 dropout rate).
3. **Flattening Layer**:
    - The output of the convolutional layers is flattened into a single vector.
4. **Fully Connected (Dense) Layers**:
    - A Dense layer with 256 neurons and ReLU activation, followed by Batch Normalization and a Dropout layer (0.5 dropout rate).
    - The final Dense layer has 7 neurons (one for each emotion class) with a softmax activation function to produce probability distributions over the emotion classes.

Transfer Learning with ResNet50

In addition to developing a custom CNN, we leveraged Transfer Learning by utilizing the ResNet50V2 pre-trained model. Transfer Learning helps accelerate the training process and improve model performance by building on the knowledge already embedded in pre-trained models.

1. **Pre-trained ResNet50V2**:
    - We used ResNet50V2, a powerful model pre-trained on the ImageNet dataset. This model has already learned to extract rich and hierarchical features from images.
2. **Fine-Tuning**:
    - The top layers of ResNet50V2 were replaced with layers specific to our task. This included a Global Average Pooling layer followed by a Dense layer with ReLU activation, Batch Normalization, Dropout, and finally a Dense layer with softmax activation for the seven emotion classes.
    - The lower layers of ResNet50V2 were frozen initially to retain the learned features and only the new top layers were trained. Gradually, more layers were unfrozen and fine-tuned with a lower learning rate to adapt the pre-trained features to our specific dataset.

By combining the custom CNN architecture with Transfer Learning from ResNet50V2, we leveraged the strengths of both approaches to create a powerful and efficient facial emotion recognition model. This dual approach enabled us to capture intricate facial features effectively while benefiting from the pre-trained knowledge of ResNet50V2, ultimately leading to improved performance and faster convergence.

Training and Evaluation

Training Process

The training phase is a critical component of the DeepFER project, where the custom CNN and ResNet50V2 models are trained on the augmented dataset. To ensure optimal performance and model robustness, we implemented various training strategies and hyperparameter optimization techniques. Here's a detailed overview of the training process:

1. **Model Compilation**:
    - ○ The models were compiled using the Adam optimizer with a learning rate of 0.0001. The loss function used was categorical cross-entropy, which is suitable for multi-class classification problems. The performance metric monitored was accuracy.
2. **Callbacks**:

- To enhance the training process, we utilized several callbacks:
    - ○ **ModelCheckpoint**: This callback saves the model weights of the best-performing model based on validation accuracy.
    - ○ **ReduceLROnPlateau**: This callback reduced the learning rate by a factor of 0.8 if the validation accuracy stopped improving for 8 consecutive epochs, preventing overfitting and enhancing model convergence.
    - ○ **EarlyStopping**: For the ResNet50V2 model, EarlyStopping was also used to halt training when validation accuracy did not improve for 7 consecutive epochs, restoring the best model weights.

3. **Training Execution**:
    - ○ The models were trained using the augmented dataset with a specified number of epochs and batch size. The fit method was used to start the training process, and the training history was recorded for further analysis.
4. **Evaluation Metrics**
    - ○ To ensure the robustness and reliability of the trained models, their performance was evaluated using several metrics on a validation set:

1. **Accuracy**:
    - ○ Measures the proportion of correctly predicted emotion classes over the total predictions.
2. **Precision, Recall, and F1-Score**:

- **Precision**: The ratio of true positive predictions to the total predicted positives.
- **Recall**: The ratio of true positive predictions to all actual positives.
- **F1-Score**: The harmonic mean of precision and recall, providing a balance between them.

These metrics were computed to ensure the model's performance was not only accurate but also balanced across different emotion classes.

During the training process, the custom CNN model achieved a training accuracy of 64.81% and a validation accuracy of 59.74%. For the ResNet50V2 model, after implementing transfer learning and fine-tuning, the validation accuracy reached 69%, demonstrating the effectiveness of leveraging pre-trained models for facial emotion recognition tasks.

In conclusion, the training and evaluation process for DeepFER involved careful model compilation, strategic use of callbacks, and comprehensive performance evaluation using multiple metrics. This approach ensured the development of a robust and reliable facial emotion recognition system, capable of delivering high accuracy and generalization across diverse facial expressions and conditions.

Real-Time Processing

Real-Time Emotion Recognition

Real-time processing is a crucial aspect of the DeepFER project, aimed at enabling the system to accurately recognize emotions from live video feeds or real-time images. This involves developing efficient algorithms that can process and classify emotions on-the-fly, ensuring prompt and accurate responses. Below is a detailed explanation of the approach and code implemented for real-time emotion recognition:

- **Setup and Model Loading**:
  - The pre-trained model, saved during the training phase as best_model.h5, is loaded into the system.
  - The model is compiled if necessary to ensure it is ready for making predictions.
- **Video Capture Initialization**:
  - A video capture object is initialized using OpenCV to access the webcam.
  - A Haar cascade classifier is loaded to detect faces within the video frames.
- **Real-Time Emotion Detection**:
  - The system continuously captures frames from the webcam.
  - Each frame is converted to grayscale to facilitate face detection.
  - Faces are detected in the grayscale frame using the Haar cascade classifier.
  - For each detected face, the following steps are performed:
    - The face region is extracted and resized to match the model's input size (48x48 pixels).
    - The resized face image is normalized and reshaped to match the model's expected input shape.
    - The pre-trained model predicts the emotion of the face.
    - The predicted emotion label is retrieved and displayed on the frame.
- **Loading and Preparing the Model**:
  - The pre-trained model is loaded and compiled if necessary, making it ready for making real-time predictions.
- **Capturing Live Video**:
  - A video capture object is initialized to access the webcam feed, allowing the system to capture live video frames continuously.
- **Face Detection**:

    - Each captured frame is processed to detect faces using a Haar cascade classifier. This step ensures that the emotion recognition model focuses on facial regions.
- **Emotion Prediction**:
    - For each detected face, the face region is resized, normalized, and reshaped to match the input requirements of the model. The model then predicts the emotion, which is subsequently displayed on the video frame.
- **Real-Time Display**:
    - The system overlays the predicted emotion labels and bounding boxes on the detected faces in real-time, providing immediate visual feedback.
- **User Interaction**:
    - The process runs continuously until the user decides to terminate it by pressing the 'q' key, ensuring a user-friendly interface for real-time emotion recognition.

By implementing these steps, DeepFER achieves real-time emotion recognition capabilities, making it suitable for various applications such as live video analysis, interactive systems, and real-time monitoring. This enhances the practical utility of the system, allowing for immediate and accurate emotion recognition from live inputs.

Application Development

Integrating Emotion Recognition into a User-Friendly Application

Developing an application for emotion recognition involves creating a seamless and intuitive interface that allows users to interact with the system effortlessly. This integration enhances the system's utility across various domains such as human-computer interaction, mental health monitoring, and customer service. Below is an explanation of the steps implemented in the Streamlit application for facial emotion recognition:

1. **Loading the Pre-trained Model**:
    - The pre-trained model, saved during the training phase as best_model.h5, is loaded and compiled to ensure it is ready for making predictions. This model has been trained to recognize seven distinct emotions from facial expressions.
2. **Defining Emotion Labels**:
    - A dictionary is defined to map the numerical output of the model to human-readable emotion labels. This ensures that the predictions can be easily understood by the user.
3. **Creating the User Interface**:
    - Streamlit, a popular framework for creating interactive web applications in Python, is used to develop the interface.
    - A file uploader component is included, allowing users to upload images in JPG, JPEG, or PNG formats. This feature makes the application accessible and easy to use.
4. **Preprocessing the Uploaded Image**:
    - The uploaded image is preprocessed to match the input requirements of the model. This involves converting the image to grayscale, resizing it to 48x48 pixels,

normalizing the pixel values, and reshaping it to the required input shape for the model.

5. **Making Predictions**:
   - Once the image is preprocessed, the model makes a prediction about the emotion displayed in the image. The predicted label is retrieved from the model's output.
6. **Displaying Results**:
   - The uploaded image and the predicted emotion label are displayed on the web interface, providing immediate feedback to the user. This visualization helps users understand the model's predictions and interact with the application effectively.

Applications and Use Cases

The developed Streamlit application can be utilized in various domains, including:

1. **Human-Computer Interaction (HCI)**:
   - Enhancing user experiences in HCI by adapting system responses based on the user's emotional state. For example, a computer or device could change its behavior if it detects that a user is frustrated or unhappy.
2. **Mental Health Monitoring**:
   - Providing support for mental health professionals by analyzing patients' emotions over time. This can help in identifying patterns or triggers for certain emotional states, assisting in diagnosis and treatment.
3. **Customer Service**:
   - Improving customer service interactions by detecting the emotional state of customers. This can enable customer service representatives to tailor their responses

By providing a user-friendly interface for emotion recognition, this application demonstrates the practical utility and versatility of the DeepFER system across multiple fields. The implementation in Streamlit ensures that the application is accessible, easy to use, and capable of delivering real-time results, making it a valuable tool for various stakeholders.

Performance Optimization

Optimizing the Model and System for Efficiency and Speed

To achieve real-time capabilities in emotion recognition, it is crucial to optimize the model and the overall system for both efficiency and speed. This involves reducing latency and computational overhead while maintaining high accuracy. Here are the key steps and strategies implemented to optimize performance:

1. **Model Architecture Optimization**:
    - **Custom CNN Design**: The custom CNN architecture is carefully designed to balance complexity and performance. By using layers like BatchNormalization and Dropout strategically, the model achieves robustness and regularization without excessive computational cost.
    - **Efficient Layers**: Using efficient layers and avoiding overly deep architectures help in reducing the time required for inference while maintaining accuracy.
2. **Transfer Learning**:
    - **Pre-trained Models**: Leveraging pre-trained models such as ResNet50V2 provides a strong foundation. These models, pre-trained on large datasets, capture rich feature representations that improve accuracy and reduce the amount of training time needed.
    - **Fine-Tuning**: Fine-tuning only the top layers of pre-trained models while freezing others reduces the training time and computational load. This approach also helps in adapting the model to the specific task of facial emotion recognition.
3. **Hyperparameter Optimization**:
    - **Learning Rate Scheduling**: Using techniques like ReduceLROnPlateau helps in dynamically adjusting the learning rate, ensuring that the model converges efficiently without overshooting the optimal solution.
    - **Batch Size and Epochs**: Experimenting with different batch sizes and number of epochs to find the optimal balance between training speed and model performance.
4. **Efficient Data Handling**:
    - **Data Augmentation**: Applying real-time data augmentation techniques such as rotation, scaling, and flipping helps in creating a more generalized model without significantly increasing training time.
    - **Data Preprocessing Pipelines**: Implementing efficient data preprocessing pipelines to ensure that the data is fed to the model quickly and without bottlenecks.
5. **Hardware Utilization**:
    - **GPU Acceleration**: Utilizing GPUs for both training and inference significantly reduces computation time. Frameworks like TensorFlow and Keras are optimized to take full advantage of GPU acceleration.
    - **Batch Processing**: Using batch processing during inference to handle multiple inputs simultaneously, thus improving throughput.
6. **Real-Time Inference Optimization**:
    - **Lightweight Models**: Developing lightweight versions of the model that can run efficiently on edge devices with limited computational resources, ensuring that the application can be deployed in real-time scenarios.
    - **Optimized Frameworks**: Using optimized deep learning frameworks and libraries that

are designed for low-latency inference.

7. **Callback Mechanisms**:
    - **Early Stopping**: Implementing EarlyStopping to halt training when no significant improvements are observed, thus saving computational resources and preventing overfitting.
    - **Model Checkpoints**: Using ModelCheckpoint to save the best performing model during training ensures that the most efficient and accurate model is used for deployment.

By focusing on performance optimization, the DeepFER system not only achieves high accuracy but also ensures that it can be deployed in practical, real-world scenarios where speed and efficiency are critical.

Conclusion

The development and deployment of the facial emotion recognition model (DeepFER) have successfully demonstrated a robust and efficient system capable of recognizing and classifying human emotions in real-time. Here are ten key points summarizing the project:

1. **Comprehensive Data Collection**:
    - A diverse dataset comprising high-quality images categorized into seven distinct emotions (angry, sad, happy, fear, neutral, disgust, and surprise) was assembled, ensuring a broad spectrum of emotional expressions.
2. **Enhanced Data Augmentation**:
    - Implementation of data augmentation techniques such as rotation, scaling, and flipping increased the dataset variability, improving model generalization and performance.
3. **Custom CNN Architecture**:
    - A tailored Convolutional Neural Network (CNN) architecture was designed, leveraging layers like BatchNormalization and Dropout for enhanced robustness and regularization.
4. **Effective Transfer Learning**:
    - Utilization of the pre-trained ResNet50V2 model, with fine-tuning of the top layers, accelerated the training process and leveraged existing knowledge for improved accuracy.
5. **Optimized Training and Evaluation**:
    - Through meticulous hyperparameter tuning, including learning rate scheduling and appropriate batch sizes, the model achieved balanced performance metrics. Training callbacks such as ReduceLROnPlateau and ModelCheckpoint ensured efficient

training and model saving.

6. **Real-Time Processing Capability**:
    - Algorithms were developed to enable real-time emotion recognition from live video feeds or images, demonstrating the model's practical applicability in dynamic environments.
7. **User-Friendly Application**:
    - Integration into a Streamlit-based application provided a user-friendly interface, making the system accessible for various domains like human-computer interaction and mental health monitoring.
8. **Performance Optimization**:
    - The model and system were optimized for efficiency and speed, reducing latency and ensuring real-time capabilities without compromising accuracy. Techniques like GPU acceleration and lightweight model versions were employed.
9. **Robust Evaluation Metrics**:
    - Comprehensive evaluation using metrics such as accuracy, precision, recall, and F1-score on validation datasets confirmed the model's robustness and reliability.
10. **Scalability and Practical Deployment**:
    - The system's design and optimization allow for scalability and deployment on edge devices, making it suitable for mobile and IoT applications. This ensures that the DeepFER model can be effectively used in real-world scenarios requiring quick and accurate emotion recognition.

Overall, the DeepFER project showcases a well-rounded approach to developing a facial emotion recognition system, emphasizing accuracy, efficiency, real-time processing, and practical deployment. This positions the system as a valuable tool for various applications in human-computer interaction and beyond.

References

1.https://coderzcolumn.com/tutorials/artificial-intelligence/scikeras-give-scikit-learn-like-api-to-your-keras-networks#5

2.https://github.com/keras-team/tf-keras/issues/42

3.https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

4.https://towardsdatascience.com/build-your-first-deep-learning-classifier-using-tensorflow-dog-breed-example-964ed0689430

5.https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/