Computer Vision Emotion Recognition:

Abstract:

This project focuses on developing an emotion recognition system capable of accurately identifying and classifying emotions from facial expressions. Utilizing computer vision and deep learning, the system analyzes facial features and categorizes them into predefined emotions such as happiness, sadness, anger, and surprise. The dataset, which was custom-built for this project, includes approximately 500+ labeled video clips, stored in the /collected_emotions/ folder, totaling around 3 GB. The model architecture, which incorporates several convolutional layers followed by dense layers, was constructed and trained using TensorFlow and Keras. The training process yielded a training accuracy of 95.2% and a validation accuracy of 91.9%, indicating strong performance in emotion classification. Once trained, this system can be used to detect and recognize emotions in real-time video feeds, making it useful for applications such as human-computer interaction, mental health assessment, and more.

Problem:

Facial expression plays a critical role in human communication, providing essential cues for understanding emotions and intentions. However, in various contexts, such as online education, remote work, and mental health assessments, accurately interpreting these expressions becomes challenging without in-person interaction. According to research, misinterpretation of emotions through digital communication channels is common, leading to misunderstandings and reduced empathy. Moreover, individuals with conditions such as autism or alexithymia often struggle to recognize and respond to facial expressions, further complicating social interactions. Given the growing reliance on digital platforms for communication, there is a pressing need for an automated system that can reliably recognize and interpret facial expressions. Such a system could enhance communication, provide support in mental health diagnostics, and improve user experiences across various digital interfaces. Therefore, I propose developing a facial expression recognition system using advanced computer vision and deep learning techniques to address these challenges.

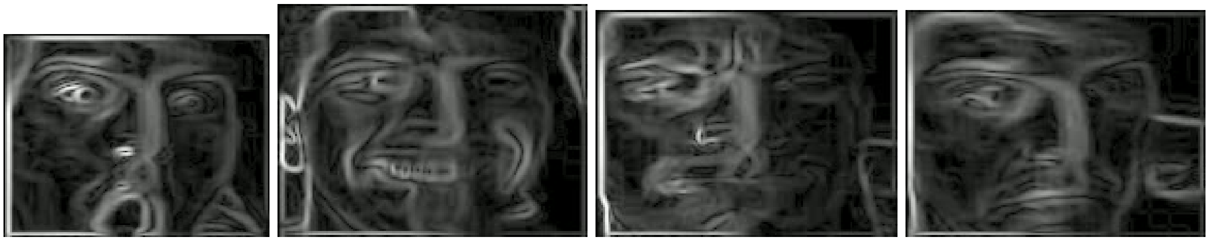
Methods:

Libraries:

TensorFlow and Keras were employed for designing, training, and fine-tuning the deep learning models, specifically the convolutional neural networks (CNNs) used for emotion classification. OpenCV was utilized for real-time image processing and handling video input, including tasks like face detection, cropping, and applying preprocessing steps such as edge detection and Gaussian blurring. The dlib, known for its high-performance face detection capabilities, was used to detect and align faces in video frames, ensuring consistency in the input data. Additionally, NumPy was used for numerical computations and array manipulations, while Matplotlib and Seaborn facilitated the visualization of data and model performance metrics throughout the training process. These libraries, combined, provided a comprehensive toolkit for developing and deploying the emotion recognition system.

Dataset:

Since there was no suitable pre-existing dataset for emotion detection, I compiled my own. I gathered around 500 video clips of people demonstrating various emotions, each labeled manually. In total, this dataset amounts to approximately 3 gigabytes. The dataset was then split into 80% training and the remaining split into testing. The ultimate size of the training and testing dataset was (396, 30, 120, 160, 3) and (99, 30, 120, 160, 3) respectively. Each video has 30 frames and is 120x160.



Dataset Creation:

1. For creating the dataset, I used OpenCV to record my face.
2. I used DLIB’s face detector model to detect and crop my face. After 30 frames had been collected the video was saved to the computer.

3. The final video was saved as a numpy array of total (495, 30, 120, 160, 3) size.

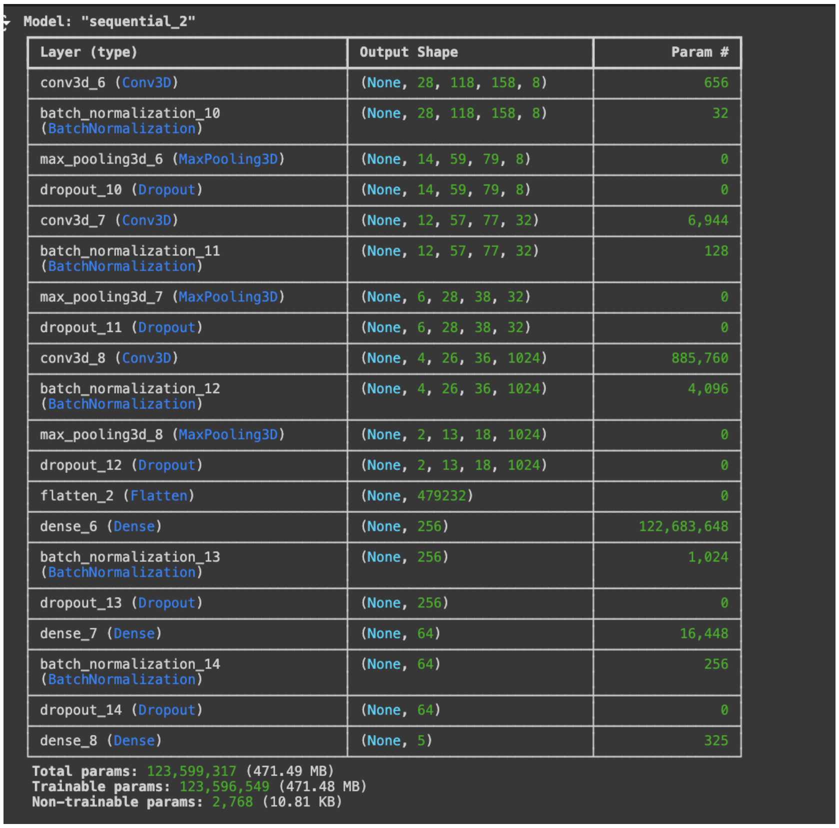
Image Preprocessing:

Preprocessing Steps

- Sobel edge detection
- Gaussian Blurring
- Random Cropping
- Gray Scaling
- Preprocessing steps were applied through literature search using the following link: Preprocessing Literature.

Sobel Edge Detection: Sobel edge detection is an image processing technique that detects edges in an image by calculating the gradient of the image intensity. It emphasizes regions of high spatial frequency, which correspond to edges.
Gaussian Blurring: Gaussian blurring smooths an image by averaging pixel values with their neighbors, weighted by a Gaussian function. This technique reduces noise and detail, often used as a pre-processing step in image analysis.
Random Cropping: Random cropping involves cutting out a random portion of an image, often used as a data augmentation technique in machine learning. It helps create variations in the training data, making models more robust to differences in image size and composition.
Gray Scaling: Grayscale conversion reduces an image's color channels to shades of gray, eliminating color information. This simplifies the image data, often used in tasks where color is not essential, like edge detection or texture analysis.

3D-CNN Model:



```
Model: "sequential_2"
┌────────────────────────────────────┬───────────────────────────┬─────────────────┐
│ Layer (type)                       │ Output Shape              │         Param # │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ conv3d_6 (Conv3D)                  │ (None, 28, 118, 158, 8)   │             656 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ batch_normalization_10             │ (None, 28, 118, 158, 8)   │              32 │
│ (BatchNormalization)               │                           │                 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ max_pooling3d_6 (MaxPooling3D)     │ (None, 14, 59, 79, 8)     │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dropout_10 (Dropout)               │ (None, 14, 59, 79, 8)     │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ conv3d_7 (Conv3D)                  │ (None, 12, 57, 77, 32)    │           6,944 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ batch_normalization_11             │ (None, 12, 57, 77, 32)    │             128 │
│ (BatchNormalization)               │                           │                 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ max_pooling3d_7 (MaxPooling3D)     │ (None, 6, 28, 38, 32)     │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dropout_11 (Dropout)               │ (None, 6, 28, 38, 32)     │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ conv3d_8 (Conv3D)                  │ (None, 4, 26, 36, 1024)   │         885,760 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ batch_normalization_12             │ (None, 4, 26, 36, 1024)   │           4,096 │
│ (BatchNormalization)               │                           │                 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ max_pooling3d_8 (MaxPooling3D)     │ (None, 2, 13, 18, 1024)   │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dropout_12 (Dropout)               │ (None, 2, 13, 18, 1024)   │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ flatten_2 (Flatten)                │ (None, 479232)            │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dense_6 (Dense)                    │ (None, 256)               │     122,683,648 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ batch_normalization_13             │ (None, 256)               │           1,024 │
│ (BatchNormalization)               │                           │                 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dropout_13 (Dropout)               │ (None, 256)               │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dense_7 (Dense)                    │ (None, 64)                │          16,448 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ batch_normalization_14             │ (None, 64)                │             256 │
│ (BatchNormalization)               │                           │                 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dropout_14 (Dropout)               │ (None, 64)                │               0 │
├────────────────────────────────────┼───────────────────────────┼─────────────────┤
│ dense_8 (Dense)                    │ (None, 5)                 │             325 │
└────────────────────────────────────┴───────────────────────────┴─────────────────┘
 Total params: 123,599,317 (471.49 MB)
 Trainable params: 123,596,549 (471.48 MB)
 Non-trainable params: 2,768 (10.81 KB)
```

After setting up the video dataset, I designed my model architecture with the intention of capturing the spatiotemporal features of the video frames. The model begins with three Conv3D layers with kernel sizes of 3x3x3 and filters of 8, 32, and 1024, respectively. Each Conv3D layer, except for the last one, is followed by a Batch Normalization layer and a MaxPooling3D layer with a pooling size of 2x2x2. This combination helps the model to normalize activations and reduce the spatial dimensions while maintaining essential features.

Dropout layers with a rate of 0.5 are added after each MaxPooling3D layer to prevent overfitting by randomly dropping a fraction of the neurons during training. This encourages the model to learn more robust features and avoid over-reliance on any specific neuron.

Following the convolutional layers, the model includes a Flatten layer that transforms the 3D feature maps into a 1D vector. This vector is then passed through a series of Dense layers, starting with 256 neurons, followed by a 64-neuron layer, and finally ending with a 5-neuron output layer, which is likely used for classification into 5 different classes. Batch Normalization and Dropout are also used after each Dense layer to further regularize the model and prevent overfitting.

In total, the model has around 123.6 million trainable parameters, which allows it to capture complex spatiotemporal patterns in the video data. The use of multiple Conv3D layers enables the model to learn features across both spatial and temporal dimensions, making it suitable for tasks such as action recognition or video classification.

Model Results:

Here are some of the metrics which I used to evaluate my project.

Metrics:

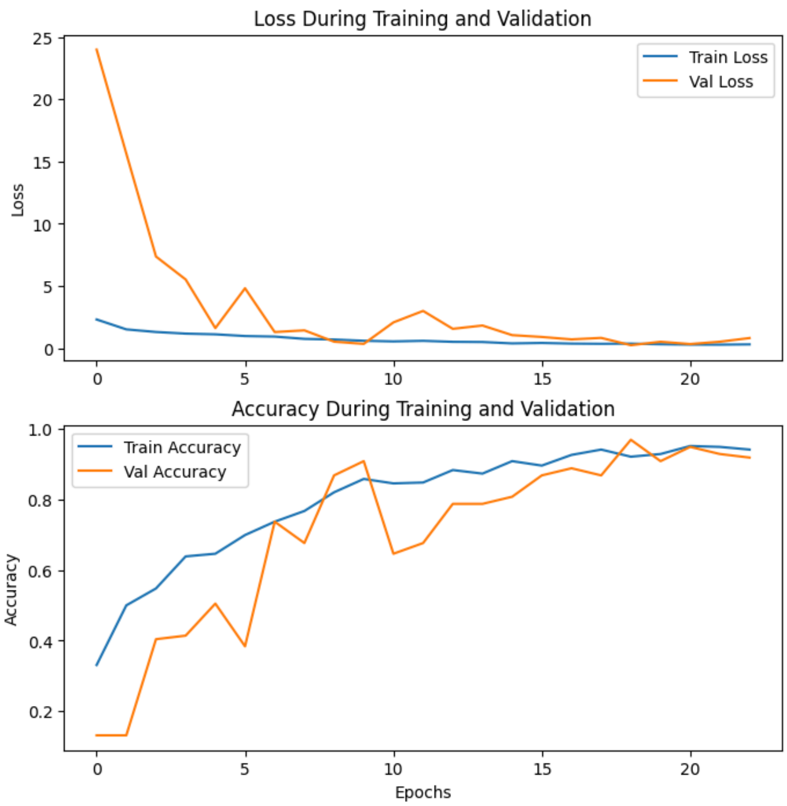
1. **Accuracy**: measures the proportion of correctly predicted instances out of the total instances. It provides a general sense of how often the model makes the right prediction but can be misleading if the dataset is imbalanced.

2. **Balanced Accuracy**: Balanced Accuracy is the average of the recall (or sensitivity) obtained on each class. It's particularly useful for imbalanced datasets as it gives equal weight to each class, regardless of how many instances belong to each class.

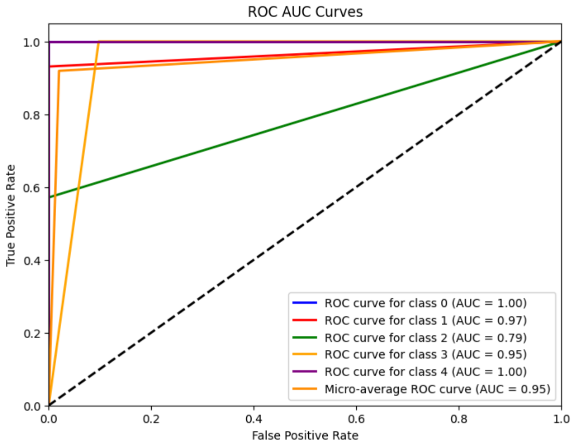
3. **Precision**: Precision is the proportion of true positive predictions out of all the predictions made for a positive class. It reflects the model's ability to not label a negative sample as positive, focusing on the relevance of the positive predictions.

4. **Recall**: Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives that were correctly identified by the model. It emphasizes the model's ability to capture all relevant instances from the dataset.

The 3D CNN was run for 27 epochs and the final training accuracy was 0.9521 and testing accuracy was 0.9192. The testing accuracy of 91.92% indicates robust performance capabilities.



The steep slope towards the top left, particularly for class 3, indicates that the model has a high true positive rate with a very low false positive rate early on, which is a sign of strong performance for this class. This steep curve suggests that the model is highly confident in correctly identifying instances of class 3 while minimizing incorrect classifications, contributing to its overall strong performance as reflected by the AUC of 0.95 for this class.



Discussion:

The results of the emotion recognition system highlight the effectiveness of using a 3D CNN architecture for analyzing spatiotemporal features in video data. Achieving a training accuracy of 95.2% and a validation accuracy of 91.9% demonstrates that the model has learned to generalize well to unseen data, indicating that the chosen preprocessing steps and model architecture were effective in capturing the nuances of facial expressions. The use of Sobel edge detection and Gaussian blurring likely enhanced the model's ability to focus on key features, while techniques like random cropping and gray scaling contributed to the robustness of the training process. However, the variation in the performance across different classes, as evidenced by the precision-recall curves, suggests that certain emotions are more easily identifiable than others. This could be due to the inherent complexity of facial expressions or the quality and diversity of the collected dataset. Despite these challenges, the overall high performance indicates that the model is well-suited for practical applications in real-time emotion recognition.

Challenges:

One of the primary challenges in this project was creating a suitable dataset for training the model. Given the lack of publicly available datasets that met the specific requirements of this project—namely, video clips with consistent labeling of emotions—building a custom dataset became necessary. Collecting and labeling around 500 video clips manually was time-consuming and required careful attention to ensure accuracy in the labels, as mislabeling could lead to poor model performance. Furthermore, ensuring that the dataset was sufficiently diverse to cover a range of facial expressions across different individuals and emotions posed an additional challenge. Balancing the dataset to avoid biases and ensuring consistency in video quality and lighting conditions were also critical factors that influenced the final model's performance.

Further Applications:

The developed emotion recognition system has the potential to be applied across various domains. In human-computer interaction, it can be used to create more responsive and empathetic systems that adjust their behavior based on the user's emotional state. For instance, virtual assistants and educational platforms could adapt their responses depending on whether the user appears frustrated or engaged. In the realm of mental health, this system could assist clinicians by providing objective assessments of a patient's emotional state over time, potentially aiding in the diagnosis of mood disorders or monitoring therapeutic progress. Additionally, in entertainment and marketing, emotion recognition can be used to analyze audience reactions to content or advertisements, enabling more personalized and effective content delivery. With further refinement and integration into existing technologies, this system could significantly enhance the way machines interpret and respond to human emotions.