Deep Learning (COSC 2779/2972) – Assignment 1 – 2023
<Harshith Hullakere Siddegowda>(<s3914263>)

Problem Definition and Analysis

The project at hand addresses the vital task of automatic facial expression detection, situated at the crossroads of computer vision, machine learning, and behavioral sciences. This undertaking involves the intricate analysis of facial images to decipher the emotional states conveyed by individuals portrayed in those images. The primary objectives of this project can be outlined as follows:

High-Level Emotion Classification: The foremost goal is to categorize facial images into three broad emotional categories: "Positive," "Negative," or "Surprised." This challenging task necessitates the training of a machine learning model capable of not only recognizing but also differentiating between these distinct emotional states.

FACS Code Detection: The second objective is the identification of specific facial actions or expressions denoted by Facial Action Coding System (FACS) codes. Each FACS code corresponds to a particular facial movement or expression. This task, however, is binary in nature. It involves determining whether a particular FACS code is present (indicated by "yes" or 1) or absent (indicated by "no" or 0).

Evaluation Framework

In the evaluation of my ResNet model, I have utilized a comprehensive framework that includes key metrics like F1 score, accuracy, precision, and recall to assess its performance in classifying emotion and FACS code outputs. Additionally, I leveraged classification reports and confusion matrices to provide a detailed view of its performance, allowing us to identify areas for model improvement. This multifaceted evaluation approach ensures a thorough understanding of the model's strengths and areas for enhancement in both emotion recognition and FACS code prediction.

Approach & Justifications

Exploratory Data Analysis (EDA) and Data Cleaning

Data Exploration: To begin the project, we performed an initial exploration of the dataset. We found that the dataset consists of three high-level emotion categories: negative (328 samples), positive (121 samples), and surprise (111 samples). This information provides an overview of the class distribution within the dataset. **Visualization of Emotion Distribution:** To gain a visual understanding of the distribution of high-level emotions, a pie chart was created. The chart clearly illustrates the distribution of emotions, with negative, positive, and surprise emotions occupying 59.6%, 21.9%, and 20.1% of the dataset, respectively. **Data Pre-processing:** During the data cleaning phase, several columns were removed from the dataset, including 'subject,' 'sequence,' 'image_index,' and 'file_prefix.' This step was taken to eliminate unnecessary information and prepare the data for further analysis.**Correlation Analysis:** A correlation matrix was computed to examine the relationships between the remaining attributes in the dataset. The resulting heatmap visually represents the correlation coefficients between variables. This analysis will help us understand the interdependencies among features and guide feature selection or engineering in subsequent stages of the project.**Data Splitting:** The dataset was divided into training (80%), validation (10%), and test (10%) sets using Scikit-Learn's train_test_split function.This split ensures adequate data for training and evaluation while maintaining a clear separation for model validation.**Creating Label Numerical Mapping:** A mapping was established to convert emotion labels (e.g., 'negative,' 'positive,' 'surprise') into numerical values (0, 1, 2), adding a 'labels_num' column to the dataset. Numerical labels are essential for machine learning models, enabling them to process emotion information effectively.

Data Loader (Data Generator class):

- The Data Generator class is a central component of our data management approach, offering memory-efficient handling of extensive datasets. The Read Data Instance method allows for individual image loading and pre-processing, granting fine-grained control over data transformations, including augmentation, normalization, and other dataset-specific operations vital for effective model training.
- Furthermore, The On Epoch End function manages index updates, reducing overfitting and promoting generalization. The Data Generation function assembles data batches, minimizing memory usage by loading only necessary data. The Get Item function efficiently provides data batches for training iterations. The output format, denoted as (56), illustrates the data structure: (8, 32, 32, 3) for input images, (8, 3) for high-level emotion labels, and (8, 15) for FACS codes.
- In practice, as exemplified by the sample output (8, 32, 32, 3) (8, 3) (8, 15), the Data Generator seamlessly generates and delivers data batches to the training loop. The first dimension '8' signifies the batch size, followed by the image dimensions (32, 32, 3) representing the input images. Subsequently, (8, 3) denotes the high-level emotion labels for each image in the batch, while (8, 15) corresponds to the FACS codes associated with the images.

Building The Model:

Establishing a Baseline Model: **VGG architecture**

I have chosen the VGG16 architecture as the baseline model for several reasons Mainly for **Proven Performance:** VGG16 has demonstrated strong performance on a wide range of computer vision tasks, including image classification and feature extraction. It has achieved competitive results on benchmark datasets, making it a reliable choice as a baseline. **ImageNet Pre-trained Weights:** VGG16 was pre-trained on the large-scale ImageNet dataset, which contains millions of images across thousands of categories. By leveraging these pre-trained weights, I can benefit from the rich features learned during this training, improving the model's ability to recognize patterns in images. **Transfer Learning:** By using VGG16 as a feature extractor, I can take advantage of the lower layers' ability to capture low-level features like edges, textures, and basic shapes. and lastly for fine-tuning capability.

Here's a breakdown of the key elements incorporated into my baseline VGG16-based model:

- **Base VGG16 Model:** I imported the VGG16 architecture with pre-trained weights from ImageNet. To maintain the integrity of the pre-trained features, I set all layers as non-trainable.
- **Shared Feature Extraction Layers:** I added custom layers on top of the VGG16 base model to adapt it for my tasks.This includes a Flatten layer to transform feature maps into a one-dimensional format, a Dense layer with 256 units and ReLU activation for feature refinement, and a Dropout layer to prevent overfitting.
- **Emotion Prediction Branch:** For emotion recognition, I added a Dense layer with 3 units and a softmax activation function. This branch produces probability distributions over three emotion classes.
- **FACS Code Prediction Branch:** Simultaneously, for FACS code prediction, I included another Dense layer with 15 units and a sigmoid activation function. This branch generates binary predictions for 15 different FACS codes.
- **Final Model Composition:** The final model is created using the Model class, specifying the input as the VGG16 model's input and the outputs as a list containing both the emotion and FACS code prediction branches.

Problem with VGG Model

In the VGG model that I have implemented, there are several key observations and problems that prompted the transition to the ResNet architecture:

1.Overfitting: The provided output from the VGG model during training indicates a clear issue of overfitting. Overfitting occurs when the model performs well on the training data but fails to generalize to unseen data (in this case, the validation data). In the mentioned output, the training accuracy for the emotion output (0.6964) is notably higher than the validation accuracy (0.1964), indicating a lack of generalization.

- The training loss (1.4051) is also lower than the validation loss (1.4969), further confirming the overfitting problem.

2. Low FACS_CODE Output Performance: The FACS_CODE output of the VGG model shows relatively low accuracy (0.3125) and precision (0.5401) during training. These metrics suggest that the model struggles to predict the FACS codes accurately.

3. Deep Network: VGG is a deep convolutional neural network (CNN) architecture with a large number of layers. While this architecture has been successful in various computer vision tasks, it can be challenging to train and prone to overfitting, especially when working with limited data.

Transition to ResNet Architecture:
The transition to the ResNet (Residual Network) architecture addresses some of the issues encountered with the VGG model: Mitigating Vanishing Gradient Problem, Enabling Deeper Networks, Better Generalization, Improved Training and Convergence. different layer that I have used in my model is,

- First Convolutional Layer (conv1): This layer performs a convolution operation on the input. It usually has a small kernel size (e.g., 3x3), which allows it to capture local features. The output of this layer is passed through batch normalization and ReLU activation.
- Second Convolutional Layer (conv2): Similar to the first convolutional layer, this layer performs another convolution operation but without changing the dimensions of the feature maps. Like the first layer, it is followed by batch normalization and ReLU activation.
- Skip Connection (down sample): This part of the block is responsible for adjusting the dimensions of the input if needed. When the stride of the first convolutional layer is not equal to 1, the skip connection is used to down sample the input to match the output dimensions. This ensures that the element-wise addition between the residual and the processed input is valid.
- Residual Addition: After passing through the conv1, conv2, and normalization layers, the output is added to the residual obtained from the skip connection. This addition is performed element-wise and allows gradients to bypass the block easily during backpropagation.
- ReLU Activation: ReLU (Rectified Linear Unit) activation is applied to the output of the entire block. This non-linearity introduces the capability to model complex relationships in the data.
- Input Layer Configuration: The input layer is defined with a shape of (32, 32, 3), indicating that the model expects input images to have a spatial resolution of 32x32 pixels and three colour channels (RGB). This input shape is compatible with the data provided and helps the model efficiently process image information.
- Emotion Output and FACS Output Layers: The emotion output layer employs a softmax activation function, which is suitable for multi-class classification tasks. It outputs probabilities for three distinct emotion classes. In contrast, the FACS output layer utilizes a sigmoid activation function. This is chosen because the FACS codes are binary labels representing the presence or absence of specific facial actions, making sigmoid activation appropriate for multi-label classification.

Experiments & Tuning

- **Batch Size Experiment**: Iam experimenting with different batch sizes of 8, 16, and 32 during training. The best training and validation accuracy were achieved with batch size 8. This suggests that smaller batch sizes allowed the model to learn more effectively, possibly by updating its weights more frequently. Smaller batches can also help the model generalize better.
- **Number of Epochs Experiment**: I have experimented with different numbers of epochs, specifically [25, 40, 50, 70, 100]. The best results were obtained when training for 40 epochs. Training for 40 epochs strikes a balance between allowing the model to learn and preventing overfitting, resulting in the best overall performance.
- **Data Augmentation**: Data augmentation was applied during training. The provided training results indicate that data augmentation had a positive impact on the model's performance. For example, Epoch 20, the model achieved good accuracy with a validation accuracy of 85.71% for emotion prediction and 44.50% for FACS code prediction. Data augmentation likely helped the model generalize better by exposing it to a variety of augmented versions of the training data, reducing overfitting and improving accuracy on unseen data.

- **Optimizer Selection**:I have considered three different **optimizers**: RMSprop, SGD (Stochastic Gradient Descent), and Adam. Each optimizer has its own way of updating model weights during training, and their effectiveness can vary depending on the dataset and problem.

  For each optimizer, I tried multiple **learning rates**: [0.01, 0.001, 0.0001]. The learning rate determines the step size taken in the weight update process during training. An optimal learning rate is crucial for efficient convergence.

  For each optimizer, I compiled the model, specifying **categorical cross-entropy loss** for the emotion output and **binary cross-entropy loss** for the FACS_CODE output.

  Best Performances:Among the optimizers tested, Adam achieved the highest F1 score for the emotion output, with a score of 0.8033. The training and validation loss tend to decrease significantly. For the FACS_CODE output, RMSprop achieved the highest F1 score, with a score of 0.4157.
- **Effect of Lambda (Regularization Strength) on Performance:** The experiments involving different lambda values (0.05, 0.01, 0.005, and 0.001) demonstrated that regularization plays a crucial role in controlling overfitting. Among the tested lambda values, 0.001 emerged as the best choice. It struck a balance between preventing overfitting during training, as evidenced by the decrease in training loss, and maintaining good generalization to the validation dataset.

Ultimate Judgment, Analysis & Limitations

- The ResNet model underwent a comprehensive evaluation process to determine its suitability for automatic facial expression detection. Metrics such as F1 score, precision, and recall were employed to assess its performance on both Emotion Output and FACS_CODE Output. The model demonstrated a strong **F1 score of 0.81** for Emotion Output, highlighting its effectiveness in identifying high-level emotions. However, the FACS_CODE Output achieved a lower F1 score of 0.49, indicating room for improvement in detecting specific facial actions by doing Data Imbalance techniques.
- Hyperparameter tuning was meticulously executed, exploring various learning rates, regularization coefficients, and batch sizes. The model's versatility is promising for real-world implementation, with its ability to accurately classify high-level emotions. Nonetheless, challenges persist in accurately recognizing nuanced facial expressions, possibly due to individual variations and dataset quality. While the model's strengths make it a compelling choice, addressing its limitations and refining its FACS_CODE Output accuracy are ongoing endeavors to harness its full potential.

Discussion on Ethical issues and biases

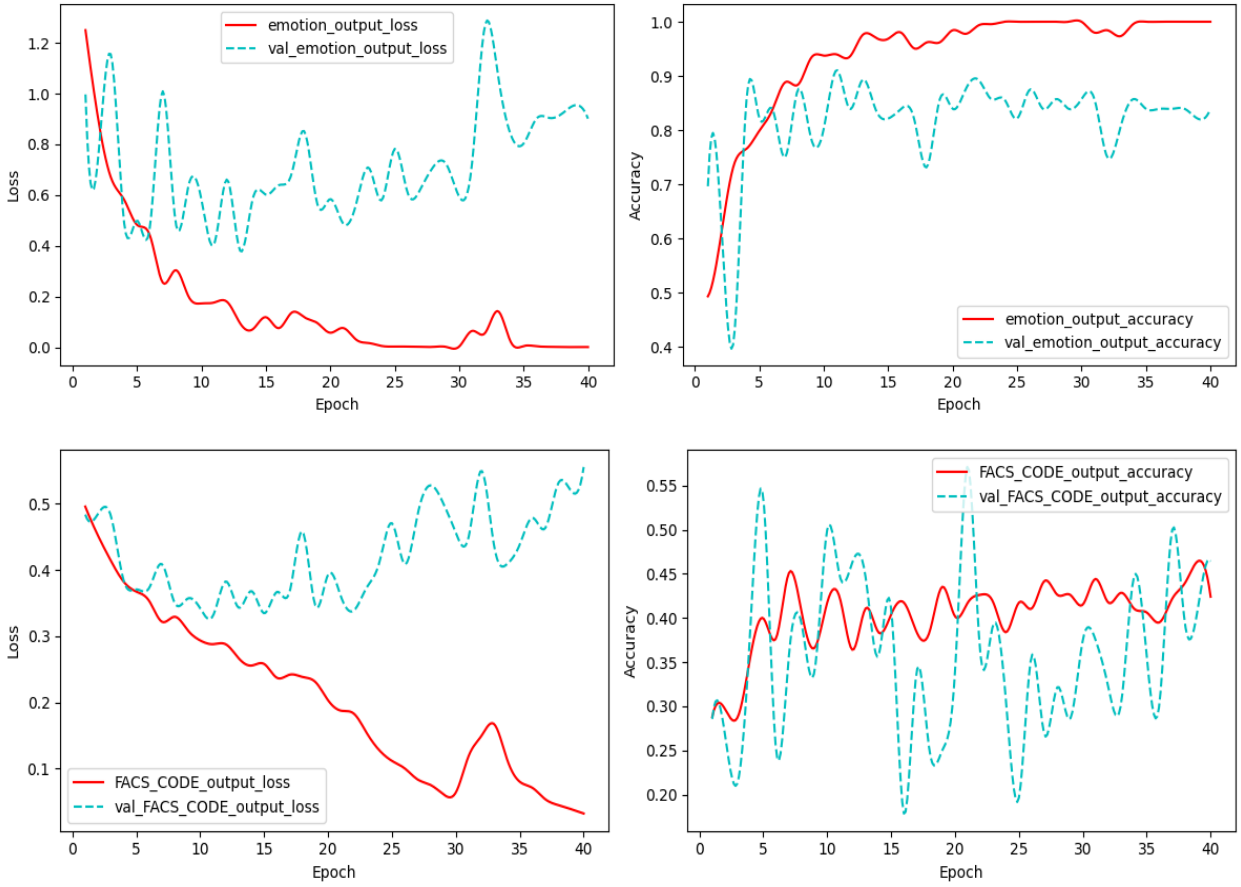
- Bias in Dataset: Biases in facial expression recognition datasets, concerning demographics, emotions, and environmental factors, can lead to inaccurate recognition and unfair treatment of specific groups.
- Bias in Algorithms: Algorithms for emotion recognition may inherit biases from training data or design choices, resulting in difficulties recognizing emotions in individuals from diverse backgrounds.
- Interpretation Bias: Even unbiased recognition models can be misinterpreted, potentially leading to unfair discrimination, particularly in contexts like job interviews.
- Privacy Concerns: Emotion recognition systems collecting personal emotional data may infringe on individuals' privacy, especially when used to infer sensitive information.
- Transparency: Informed consent for emotion data collection and usage is essential to ensure individuals understand how their emotional data is being utilized.
- Algorithm Auditing: Regular evaluation of algorithm fairness across demographic groups is crucial to prevent discrimination.

References:

[1]P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression." Available: http://www.jeffcohn.net/wp-content/uploads/2020/02/CVPR2010_CK2.pdf.pdf

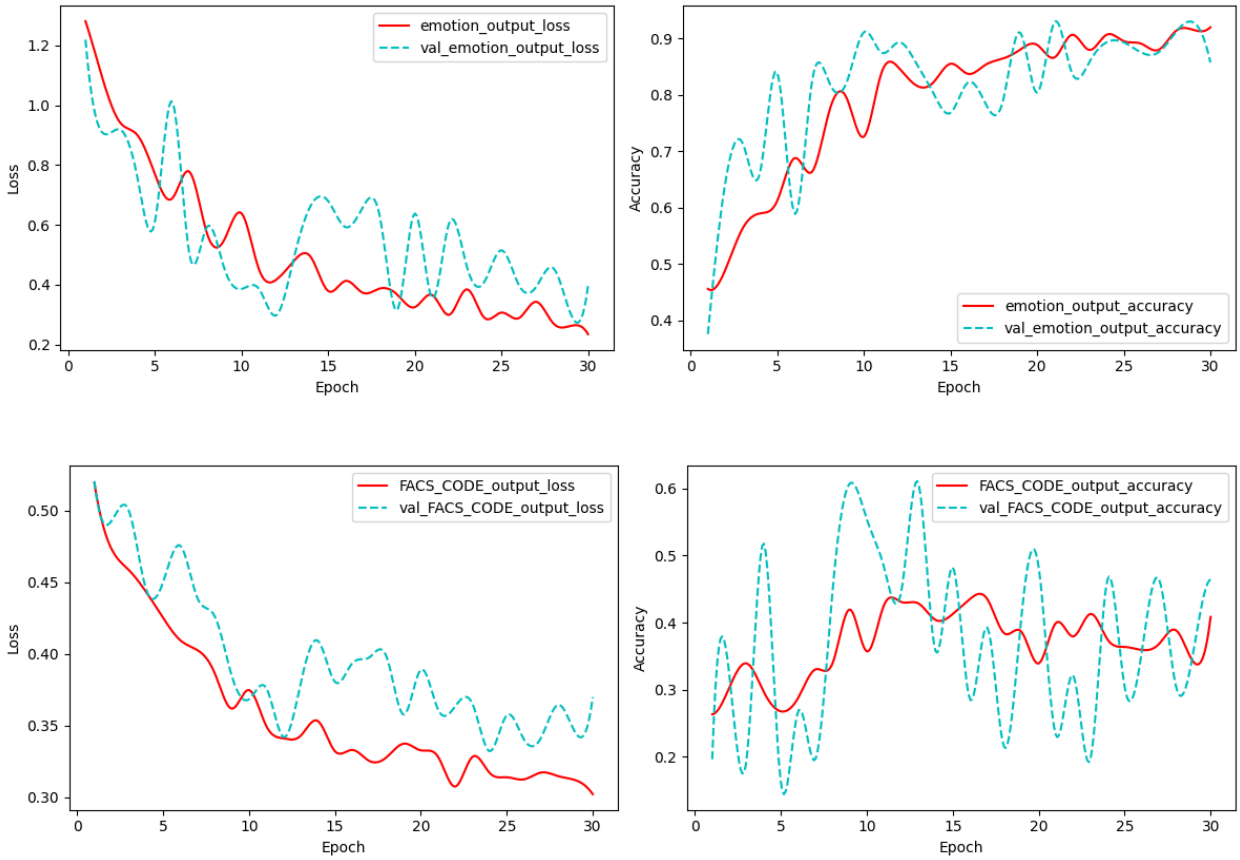
Appendices

Resnet model: **Before Hyperparameter Tuning** Figures1



Evaluation Frame	VGG Model	ResNet Model (Custom Layer)
Training Emotion Output Accuracy	0.97	0.92
Val Emotion Output Training Accuracy	0.69	0.90
Training FACS Code output Accuracy	0.40	0.47
Val FACS Code output	0.29	0.46

ResNet ModelAfter Hyperparameter Tuning: Figures 2



Metrics	Emotion Output	FACS_CODE Output
F1 Score	0.81	0.46
Precision	0.80	0.48
Recall	0.82	0.46

Classification Report for Emotion Output:

	precision	recall	f1-score	support
0	0.82	0.92	0.87	36
1	0.57	0.36	0.44	11
2	1.00	1.00	1.00	9
accuracy			0.82	56
macro avg	0.80	0.76	0.77	56
weighted avg	0.80	0.82	0.81	56

Classification Report for FACS_CODE Output

	precision	recall	f1-score	support
0	0.90	0.75	0.82	24
1	0.00	0.00	0.00	13
2	0.00	0.00	0.00	0
3	0.33	0.46	0.39	13
5	0.00	0.00	0.00	0
8	0.00	0.00	0.00	3
10	0.29	0.67	0.40	3
12	0.00	0.00	0.00	0
accuracy			0.46	56
macro avg	0.19	0.23	0.20	56
weighted avg	0.48	0.46	0.46	56