Project: checkpoint 1

1) Data Preprocessing and Dataset Statistics

The dataset is initially preprocessed, and its statistics are analyzed as:

- **Dataset Preparation**: Image files from both training and testing directories (/content/dataset/train and /content/dataset/test) are loaded. Images are handled based on their dimensionality, ensuring they are properly formatted as 3-channel RGB images even if they are originally in grayscale or have an alpha channel.
- **Normalization**: Pixel values of images are normalized to the range [0, 1] by dividing by 255. This step helps in reducing the variance within the data which can lead to better performance and stability during training.
- **Label Handling**: Categories are retrieved from the directory names, sorted and used to create a label dictionary that maps category names to integer labels.
- **Statistics**: The total number of training images is 28,709, and testing images is 7,178. The dataset supports 7 unique classes ('angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'). All images have a unified size of (48, 48) pixels.

2) Converting, Training and Normalization

- **Label Conversion**: Labels are one-hot encoded to fit the output layer of the neural network. This transformation is essential for multi-class classification with a categorical cross-entropy loss function.
- **Data Splitting**: The training dataset is split into training and validation sets with 20% of the data reserved for validation. This allows the model to be evaluated on unseen data, helping gauge its generalization capability.
- **ImageDataGenerator**: Utilized for data augmentation to artificially expand the training dataset by applying random transformations. This helps prevent overfitting and allows the model to learn more robust features. Transformations include rotations, width/height shifts, shearing, zooming, and horizontal flips.

3) Implementation of VGG Model

- **Model Architecture**: The VGG-13 model is implemented with several convolutional blocks, each consisting of Conv2D layers with ReLU activation, Batch

Normalization, and MaxPooling2D layers to reduce spatial dimensions progressively.
- **Regularization**: L2 regularization is applied to all convolutional and dense layers to penalize large weights and prevent overfitting.

4) Description of the CNN Model

The VGG-13 architecture defined includes:

- **Convolutional Blocks**: Each block is designed to capture complex patterns in the data, with deeper blocks capturing higher-level features. Batch normalization is used post-activation to standardize inputs to layers within each block.
- **Fully Connected Layers**: After flattening the output from the convolutional base, two dense layers with 1024 units each are used to classify the features extracted by the convolutional base. Dropout is included after each dense layer with a rate of 0.5 to further combat overfitting by randomly dropping units during training.

5) Impact of Techniques on Model Performance

- **Regularization (L2)**: Helps manage model complexity, discouraging large weights through penalties, which in turn helps reduce overfitting but can also lead to underfitting if too strong. It's crucial in a large model like VGG-13, which has a significant number of trainable parameters.
- **Dropout**: Randomly dropping units (and their connections) during training introduces noise into the output of hidden layers, which helps prevent neurons from co-adapting too much. This technique is very effective in large neural networks to improve generalization.
- **Early Stopping**: Monitors the model's performance on a validation set and stops training once the performance ceases to improve, preventing overfitting and reducing computational waste. This is typically used in conjunction with a model checkpointing system that saves the best version of the model throughout the training process.

These techniques collectively help in enhancing the model's ability to not only learn generalizable patterns but also prevent overfitting, which is crucial for maintaining good performance on new, unseen data.

References:

A1 assignment code.

https://arxiv.org/pdf/1702.05373

https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html

https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

https://www.projectpro.io/article/facial-emotion-recognition-project-using-cnn-with-source-code/570