**Pattern Lab Assignment 2**

Name: Pranesh Chowdhury

ID: 202003112

Section: 03

Course: CSE 436 – Pattern Recognition Lab

**1. How have you pre-processed the dataset? (EDA, explain train\_test split, data augmentation etc.)**

First I check the dataset to understand how different classes are distributed and how many images we have for each class. This process is called exploratory data analysis (EDA). After that, I divided the dataset into two parts: 80% for training our model and 20% for testing its performance.

To make our model more robust, I used a technique called ImageDataGenerator. This tool helped us create variations of our images on-the-fly during training. For example, it rotated, zoomed, and flipped images, which helped our model become better at recognizing patterns and performing well on new, unseen data.

**2. State your hyper parameters. How did you fine tune the model?**

Now check the hyper parameters, like learning rates, batch sizes, and epoch counts to get the best performance from our model. I carefully chose these hyper parameters by using methods like grid search, where I systematically tested different combinations to find the optimal values.

Cross-validation was important because it allowed me to evaluate how well my model performed in different situations with varying hyper parameter configurations. This ensured that my model was robust and could handle diverse scenarios effectively. Essentially, cross-validation helped me confirm that the model's performance wasn't dependent on specific settings and could generalize well to different conditions.

**3. Explain your classification result for each model. Were there any over fitting issues? Provide evidence.**

We used different measures like accuracy, precision, recall, and F1-score to figure out how well our models were doing in classifying things. To understand if our model was learning too much from the training data and not performing well on new data, we looked at graphs showing the loss and accuracy during training and validation. If the training numbers kept getting better but the validation numbers stayed the same or got worse, it was a sign our model might be memorizing the training data too much, and not really learning how to handle new, unseen data.

Here I used CNN, VGG16, ResNet34, Inception V3 and Transfer Learning.

For CNN model the validation graph shows the under fitting issue.

**CNN model:**

The Simple CNN, or Convolutional Neural Network, comprises three convolutional layers followed by max-pooling layers. These convolutional layers are responsible for extracting features from the input images, and the subsequent max-pooling layers downsample the spatial dimensions. The architecture is then flattened, and the feature representation is fed into dense layers with dropout, enhancing the model's ability to generalize by preventing overfitting during training. Finally, the dense layers produce the output for classification.

**VGG-16:**

The VGG-16 model implementation adheres to the VGG-16 architecture, featuring convolutional and max-pooling layers. VGG-16 is characterized by its simplicity, consisting of repeated 3x3 convolutional layers with max-pooling in between. This design enables effective feature extraction. Following the convolutional layers, dense layers are employed for classification, providing a comprehensive understanding of the extracted features. The VGG-16 model is known for its straightforward yet powerful structure.

**ResNet-34 Model:**

In the ResNet-34 implementation, residual blocks are defined and utilized to build the ResNet-34 architecture. These residual blocks introduce skip connections, allowing the network to learn residual mappings effectively. This design mitigates the vanishing gradient problem, enabling the training of deep neural networks. The ResNet-34 model specifically incorporates residual blocks of varying complexities, contributing to its depth. The model's architecture promotes efficient feature learning and facilitates the training of deep networks.

**InceptionV3:**

The InceptionV3 model adopts a different approach by leveraging a pre-trained InceptionV3 base model without the top layers. The base model, trained on large-scale image datasets, serves as a potent feature extractor. Global Average Pooling is then applied to reduce the spatial dimensions, followed by additional dense layers for classification. The utilization of Global Average Pooling captures essential features while significantly reducing the number of parameters, enhancing model efficiency. This approach is particularly beneficial when dealing with limited computational resources.

**Transfer Learning:**

In the Transfer Learning with VGG16, the pre-trained VGG16 model, equipped with weights learned from a large dataset, is employed. To leverage this pre-trained knowledge, the layers of the VGG16 model are frozen, preventing further weight updates during training. Custom dense layers are subsequently added on top to adapt the model for the specific classification task at hand. This transfer learning approach allows the model to capitalize on the knowledge gained from a diverse dataset, making it particularly useful when the available dataset for the current task is limited.

**4. Which model has performed the best in your case, and what factors contributed to its strong performance?**

To find the best model, we carefully looked at how well each one balanced complexity and the ability to handle new information. We considered the type of data we had and the specific task we were working on. Since we didn't have clear-cut metrics, we closely examined graphs showing how well the models learned during training and validation. This helped us compare and decide which model performed the best for our specific needs.