INFO 6147 - DEEP LEARNING WITH PYTORCH PROJECT REPORT

TOPIC - DISEASE PREDICTION FOR SUGARCANE PLANT LEAVES USING IMAGE CLASSIFICATION TECHNIQUES

- **INTRODUCTION:**

This project presents methods like **Image Classification** to detect, quantify and predict plant diseases. Although disease symptoms can manifest in any part of the plant, only methods that explore visible symptoms in **leaves** will be considered. Crop diseases are a major threat to food security, but their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. Sugarcane is one of the most important agricultural crops in the world. Sugarcane is a **long durational crop** due to this it is prone to more diseases. This venture is to observe the effectiveness of Image Classification techniques for the detection of illnesses in sugarcane plants by way of the use of Classification Models (like **Convolutional Neural Networks (CNN)** to distinguish unique plant diseases).

- **PROJECT OVERVIEW:**

This report compares the performance of two ResNet-18 models trained on a proprietary image dataset. The dataset, created and owned exclusively by me, is designed to classify images into nine distinct classes. The models are evaluated based on their ability to accurately classify these images. The first model is trained without hyperparameter tuning, while the second model undergoes tuning to optimize performance.

- **Dataset**:
  - Total size: ~480 MB.
  - Images: Subfolders for each class.
  - Classes: 9 (with imbalanced data—minority classes had only 8, 10, and 14 images).
  - Class names: Nitrogen Deficit, Leaf Mite, Leaf Scorch, Common Rust, Healthy, Eyespot, Mosaic Virus, Red Rot, Potassium Deficit
  - Dataset Link

- **Tools & Libraries**: PyTorch, Google Colab (GPU), Matplotlib for visualization.

- **MODEL ARCHITECTURE:**

Both models use the ResNet-18 architecture with pretrained weights. In the first model, the model's weights are frozen except for the last fully connected layer, which is modified to match the number of classes in the dataset (9 classes). In the second model, similar architecture is used, but hyperparameter tuning is performed on learning rate and optimizer choice.

- **HYPERPARAMETER TUNING:**

1. **Model 1 (Initial Training):**

   - Optimizer: Adam
   - Learning Rate: 0.001
   - Epochs: 10

2. **Model 2 (After Hyperparameter Tuning):**

   - Optimizer: SGD with momentum
   - Learning Rate: 0.01 (with learning rate scheduler)
   - Epochs: 5
   - Scheduler: StepLR (learning rate decay by a factor of 10 every 2 epochs)

- **PERFORMANCE MATRIX:**

Both models were evaluated on the test set and their performance was measured using accuracy. The following results were observed:

- **Model 1 (Without Hyperparameter Tuning):** Test Accuracy: **92.86%**
- **Model 2 (With Hyperparameter Tuning):** Test Accuracy: **95.24%**

- **TRAINING AND VALIDATION LOSS/ACCURACY GRAPHS:**

Graphs were plotted to visualize the training and validation loss and accuracy for both models over their respective epochs.

- **Model 1** shows the loss decreasing gradually with the accuracy increasing. However, there is a slight fluctuation in the validation accuracy, indicating that the model is slightly overfitting on the training set.
- **Model 2**, after hyperparameter tuning, shows a smoother curve for both loss and accuracy, indicating better convergence and improved model generalization.

- **CONCLUSION:**

The second model, after undergoing hyperparameter tuning, demonstrated a significant improvement in test accuracy, increasing from 92.86% to 95.24%. The optimization of the learning rate and use of a learning rate scheduler helped the model converge more effectively, leading to better generalization and performance on the test set.