****

**The University of Faisalabad**

**Final Project Documentation**

**BS Artificial Intelligence**

**Submitted by:**

Aneeq Kashif (2022-BSAI-022)

Saram Aamir (2022-BSAI-029)

**Submitted to:**

Ms Nida Hussain

**Subject:**

Programming for AI

**Date:**

10th June,2024

**PLANT DISEASE Detection**

**Project Documentation**

**Table of Contents**

* **Introduction**
* **Project Setup**
* **Imports**
* **Image Data Preparation**
* **Model Architecture**
* **Model Compilation**
* **Data Generators for Validation**
* **Model Training and Saving**
* **Model Loading and Testing**
* **Conclusion**

**1. Introduction**

* This project involves building an image classification model to detect plant diseases using transfer learning with the VGG19 architecture. Transfer learning leverages pre-trained models on large datasets (like ImageNet) and fine-tunes them for specific tasks, saving time and computational resources while improving accuracy.

**2. Project Setup**

**2.1 Environment Configuration**

* Project is designed to be run in a Google Colab environment, which provides the necessary computational resources and ease of use for running complex machine learning models.

**2.2 Installing Dependencies**

* Ensure you have the following Python libraries installed:
  + Tensorflow
  + Keras
  + Os

**3. Imports**

* We import necessary libraries and modules like Keras, TensorFlow, and the required layers and utilities.

**4. Image Data Preparation**

* We define the size of the images (IMAGE\_SIZE) that we will use for training and testing.
* We set up an ImageDataGenerator for preprocessing the images. This includes rescaling pixel values, shearing, zooming, and horizontal flipping for data augmentation.
* We create a data generator (train\_generator) for loading and preprocessing images from the training directory. This generator will yield batches of images along with their labels.

**5. Model Architecture**

* We load the VGG19 model with pre-trained ImageNet weights and exclude the top (classification) layers.
* We freeze all layers in the pre-trained VGG19 model so that they won't be trained again during our training process.
* We flatten the output of the VGG19 model.
* We add a dense (fully connected) layer with softmax activation to perform multi-class classification. The number of units in this layer corresponds to the number of classes in our dataset.

**6. Model Compilation**

* We compile the model, specifying the optimizer (Adam), loss function (categorical cross entropy for multi-class classification), and evaluation metric (accuracy).

**7. Data Generators for Validation**

*  We set up another data generator (val\_generator) for validation data. This generator will be used to evaluate the model's performance on a separate validation set.

**8. Model Training and saving**

* We train the model using the fit method, passing the training generator, number of steps per epoch (restricting to 10 here), number of epochs, validation data generator, validation steps, and the custom checkpoint callback.
* After training, we save the final trained model to a specified path (model\_path). If the directory does not exist, it will be created.

**9. Model Loading and Testing**

* We check if the saved model file exists and load it back into memory.
* Finally, we print a success message if the model is loaded successfully, otherwise, we print an error message.

**10. Conclusion**

* The code is able to detect plant disease with a great accuracy with a score of 2.07 and loss less than 0.5