Plant Classification on MalayaKew Dataset using a custom CNN Architecture: A Comprehensive Evaluation

Introduction:

Background:

Deep Learning has emerged as an extremely powerful tool for various tasks involving computer vision, natural language processing, etc. One of the most promising DL architectures is CNN (Convolutional Neural Networks) which demonstrates remarkable performance in various classification and identification tasks in various different fields.

Research Problem & Objectives:

The MalayaKew dataset is a widely used dataset for various tasks involving plant identification and CNNs, this dataset involves images of 44 classes of different plant species, which are split into test and train folders for training and testing your model. However, despite its importance, there's a lack of researches and CNNs done on this specific dataset that thoroughly examine and evaluate the performance of CNNs on this dataset.

In this study I plan on conducting a thorough evaluation of a novel CNN architecture on the MalayaKew Dataset. The objectives are:

1. Evaluate the performance of a custom CNN model on the MalayaKew dataset
2. Use Data Augmentation and measure the difference in performance
3. Tuning hyper-parameters and layers to generate best results possible
4. Analyz the feature extraction capabilities of the CNN model on the unseen data (test)

Significance of the Study:

The results of this study should provide a decent insight into the capabilities of this specific architecture in analyzing features of plants and carrying out plant identification, as well as give me experience on how different models perform on this specific dataset.

Methodology:

Layer	Type	Activation Function	Parameters
Input	Convolutional	None	-
Conv1	Convolutional (3x3)	ReLU	32 filters
MaxPooling1	MaxPooling (2x2)	None	-
Conv2	Convolutional (3x3)	ReLU	64 filters
MaxPooling2	MaxPooling (2x2)	None	-
Flatten	Flatten	None	-
Dense1	Fully Connected	ReLU	128 units
Dropout1	Dropout (0.5)	None	-
Dense2	Fully Connected	ReLU	64 units
Dropout2	Dropout (0.5)	None	-
Output	Softmax	None	2 units

Pre-Processing & Data Augmentation:

Before training the CNN model, I did some pre-processing to the images in the data set, first we normalized the pixel values to a range between 0 and 1, this helps improve convergence of the model.

The techniques I preferred to use on the data during Augmentation:

Random Horizontal Flipping, Random Vertical Flipping, Rescaling,

Random Rotation and Zoom.

Hyperparameter Tuning:

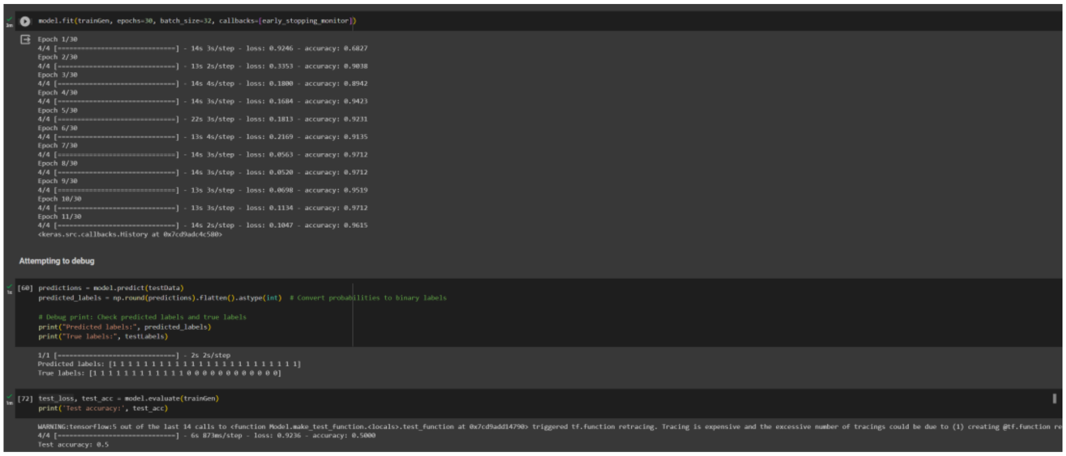
To optimize the performance of the CNN model, we attempted to tune different Hyperparameter in order to find the best suitable tune for our CNN, those different combinations that we explored included the following hyperparameters: E-pochs, Droppout rate, Layer architecture, etc.

Dataset:

The MalayaKew Plant Leaf Dataset is a publicly available dataset containing images of leaves from 44 different plant species. The dataset consists of scan-like images of leaves with size 256 * 256 pixels, providing a consistent and well-lit representation of the leaf samples. It is divided into a training set (2288 images) and a test set (528 images), enabling effective model evaluation.

Overall, the MalayaKew Plant Leaf Dataset is a valuable resource for plant classification research due to its size, diversity, and well-annotated nature. The preprocessing steps applied to ensure data quality further enhance its reliability for model development and evaluation. Its size is around 430MB.

Results:



```
model.fit(trainGen, epochs=30, batch_size=32, callbacks=[early_stopping_monitor])
```

```
Epoch 1/30
4/4 [==============================] - 14s 3s/step - loss: 0.9246 - accuracy: 0.6827
Epoch 2/30
4/4 [==============================] - 13s 2s/step - loss: 0.3353 - accuracy: 0.9038
Epoch 3/30
4/4 [==============================] - 14s 4s/step - loss: 0.1800 - accuracy: 0.8942
Epoch 4/30
4/4 [==============================] - 14s 3s/step - loss: 0.1684 - accuracy: 0.9423
Epoch 5/30
4/4 [==============================] - 22s 3s/step - loss: 0.1813 - accuracy: 0.9231
Epoch 6/30
4/4 [==============================] - 13s 4s/step - loss: 0.2169 - accuracy: 0.9135
Epoch 7/30
4/4 [==============================] - 14s 3s/step - loss: 0.0563 - accuracy: 0.9712
Epoch 8/30
4/4 [==============================] - 14s 3s/step - loss: 0.0520 - accuracy: 0.9712
Epoch 9/30
4/4 [==============================] - 13s 3s/step - loss: 0.0698 - accuracy: 0.9519
Epoch 10/30
4/4 [==============================] - 13s 3s/step - loss: 0.1134 - accuracy: 0.9712
Epoch 11/30
4/4 [==============================] - 14s 2s/step - loss: 0.1047 - accuracy: 0.9615
<keras.src.callbacks.History at 0x7cd9adc4c580>
```

Attempting to debug

```
[60] predictions = model.predict(testData)
     predicted_labels = np.round(predictions).flatten().astype(int)  # Convert probabilities to binary labels

     # Debug print: Check predicted labels and true labels
     print("Predicted labels:", predicted_labels)
     print("True labels:", testLabels)
```

```
1/1 [==============================] - 2s 2s/step
Predicted labels: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
True labels: [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
[72] test_loss, test_acc = model.evaluate(trainGen)
     print('Test accuracy:', test_acc)
```

```
WARNING:tensorflow:5 out of the last 14 calls to <function Model.make_test_function.<locals>.test_function at 0x7cd9add14790> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function re
4/4 [==============================] - 6s 873ms/step - loss: 0.9236 - accuracy: 0.5000
Test accuracy: 0.5
```

From the previous screenshots you can see that the model is not functioning correctly, I could not exactly tell the reason why during the time allowed for the research, however, I attempted to debug my model and it seems as if it's not learning, it's producing all predictions as 1 so either one class is overwhelming the other class or the model's architecture is not exactly correct.

References:

https://www.kaggle.com/datasets/abdulhasibuddin/malayakew-plant-leaf-dataset/ - MalayaKew from Kaggle

https://bard.google.com/ - Guidance from Bard

https://chat.openai.com/ - Guidance from ChatGPT-3