CSAI 867 Project 1

Adham Mokhtar - 20398545

Project Description:

The objective of this playground competition is to use binary leaf images and extracted features, including shape, margin & texture, to accurately identify 99 species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species. They also provide a fun introduction to applying techniques that involve image-based features.

Dataset Description:

The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species) which have been converted to binary black leaves against white backgrounds. Three sets of features are also provided per image: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample.

Note that of the original 100 species, we have eliminated one on account of incomplete associated data in the original dataset.

[6]:
```
df.head()
```

[6]:

	id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	...	texture55	texture56	texture57	texture58	texture59	texture60	texture61	texture62	texture63	texture64
0	1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.0	...	0.007812	0.000000	0.002930	0.002930	0.035156	0.0	0.0	0.004883	0.000000	0.025391
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.0	...	0.000977	0.000000	0.000000	0.000977	0.023438	0.0	0.0	0.000977	0.039062	0.022461
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.0	...	0.154300	0.000000	0.005859	0.000977	0.007812	0.0	0.0	0.000000	0.020508	0.002930
3	5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.0	...	0.000000	0.000977	0.000000	0.000000	0.020508	0.0	0.0	0.017578	0.000000	0.047852
4	6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.0	...	0.096680	0.000000	0.021484	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.031250

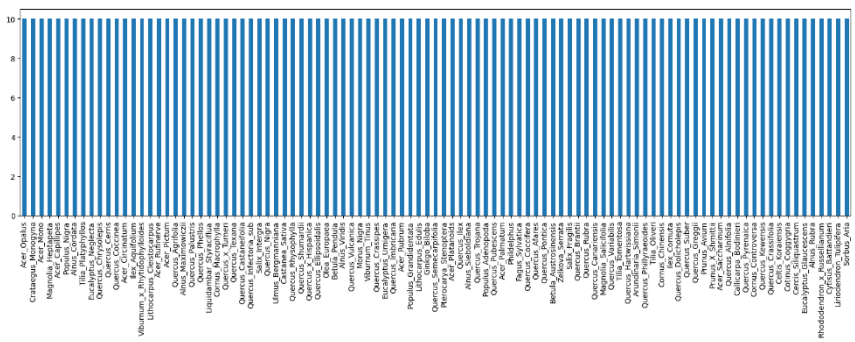
5 rows × 194 columns

After searching and exploring the data, it is obvious that the data is clean:

- Zero nulls
- Zero missing values
- Zero duplicated rows

Data consist of 99 Class as shown here:

Each class has only 10 samples.



Hyperparameter tuning:

I tried different values of hyperparameters:

- Batch size.
- Hidden size.
- Dropout.
- Optimizer.
- Regularization (weight decay).
- Learning rate.

I tired all the hyperparameter with Adam Optimizer for best results.

Here I will discuss only:

- Batch Size
- Hidden Size
- Dropout
- Learning Rate

After test and trails I figured that the best hyperparameters from the ones I had tested:

- Batch Size: 64
- Hidden Size: 1024
- Dropout Rate: 0.3
- Learning Rate: 0.001

In the next section I will show:

- The hyperparameter values that been tested.
- Accuracy and Loss graphs over the training process to show the progress of the model.
- Table of the best Accuracies and Losses
    - Training
    - Validation
    - Testing

1- Batch Size Trails:

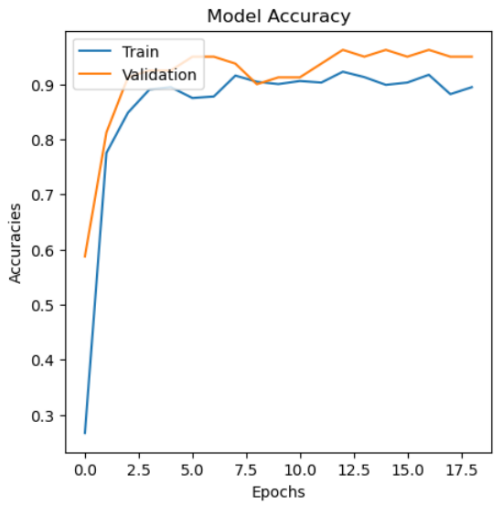
I tired four values [16, 32, 64, 128]:

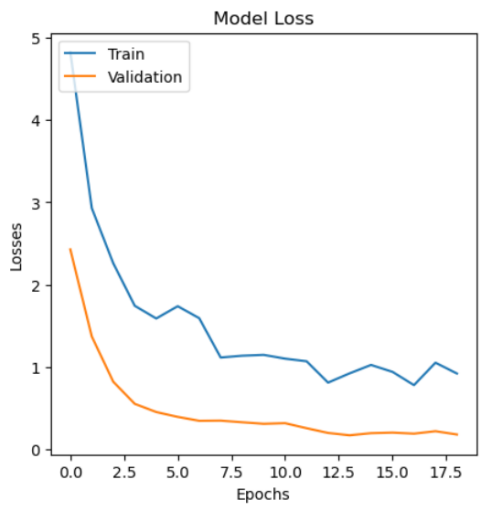
```
batch_sizes = [16, 32, 64, 128]
print(f'Start Batch Size Trails of values: {batch_sizes}')
for batch_size in batch_sizes:
    print('**********************************************************************************************')
    print(f'Start of Batch Size: {batch_size}, Trail')
    print()
    model = create_model_Adam(hidden = 256, with_dropout = True, dropout_rate = 0.1, lr = 0.001, l2 = None)
    print()
    history = model.fit(X_train, y_train, batch_size = batch_size, epochs = 50, validation_split = 0.1, callbacks=[callback])
    print()
    display_accuracy(history)
    print()
    display_val_accuracy(history)
    print()
    evaluate(model)
    print()
    print(f'End of Trail')
    print('**********************************************************************************************')
```

I observe that:

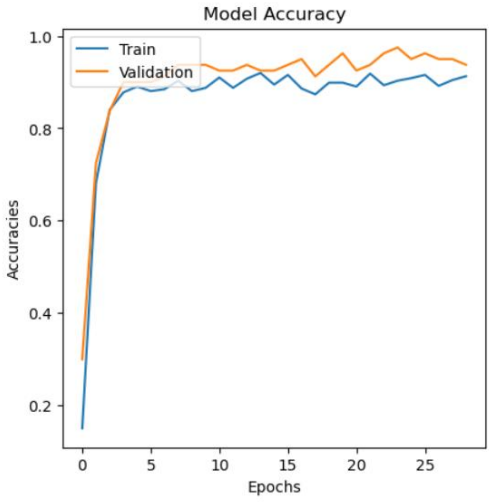
Batch Size	Loss	Accuracy	Val Loss	Val Acc	Test Loss	Test Acc
16	0.9194	0.9129	0.1660	0.9500	4.1035	0.5707
32	0.9160	0.9031	0.1660	0.9750	4.0900	0.5808
64	1.2947	0.8862	0.2249	0.9625	4.2214	0.5909
128	1.0913	0.9045	0.2280	0.9375	4.2447	0.3686

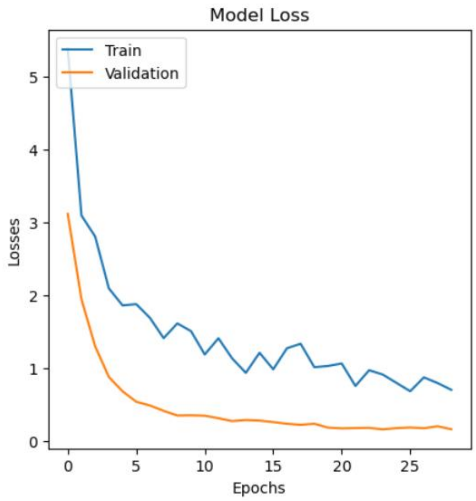
1. Batch size = 16



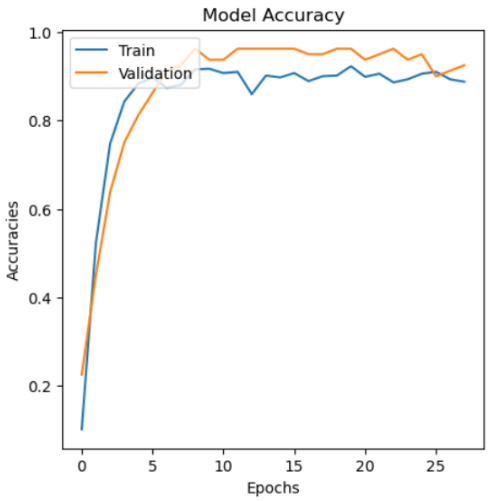


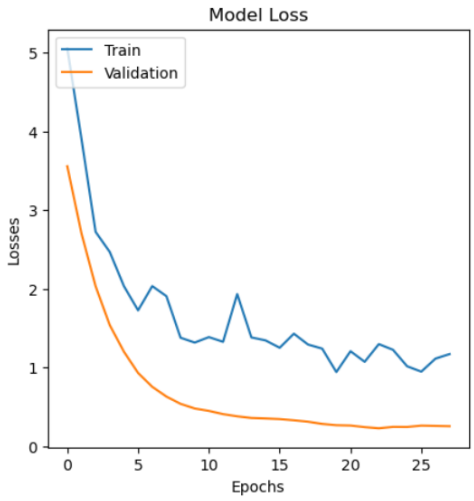
2. Batch size = 32



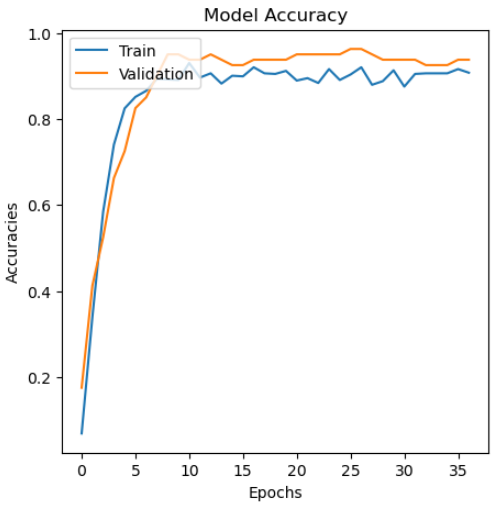


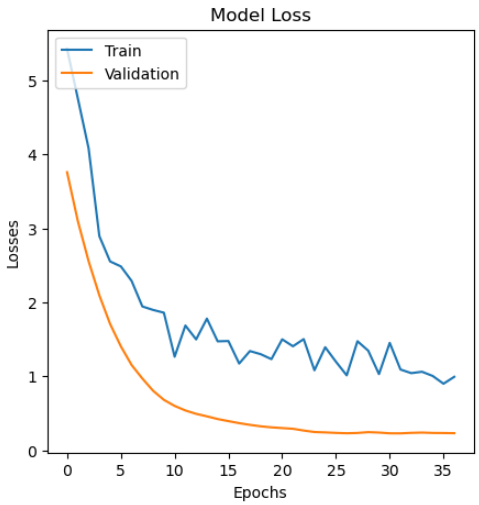
4. Batch size = 64





3. Batch size = 128





2- Hidden Sizes Trails:

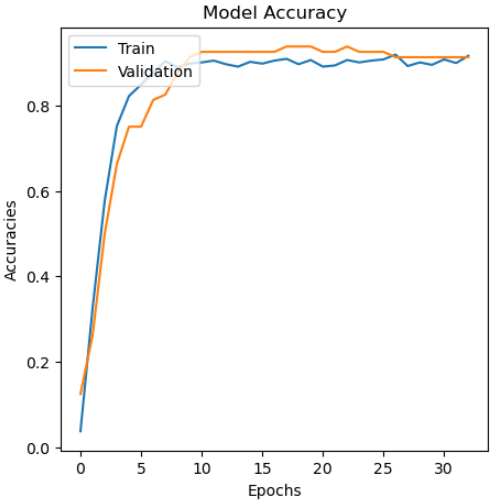
I tired three values [256, 512, 1024]:

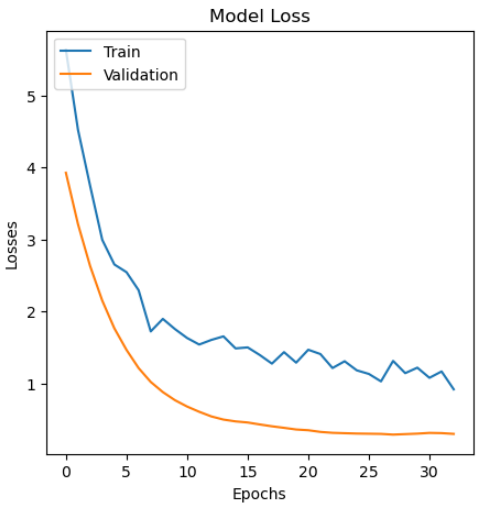
```
hiddens = [256, 512, 1024]
print(f'Start Hidden Size Trails of values: {hiddens}')
for hidden in hiddens:
    print('**********************************************************************************************')
    print(f'Start of Hidden Size: {hidden}, Trail')
    print()
    model = create_model_Adam(hidden = hidden, with_dropout = True, dropout_rate = 0.1, lr = 0.001, l2 = None)
    print()
    history = model.fit(X_train, y_train, batch_size = 128, epochs = 50, validation_split = 0.1, callbacks=[callback])
    print()
    display_accuracy(history)
    print()
    display_val_accuracy(history)
    print()
    evaluate(model)
    print()
    print(f'End of Trail')
    print('**********************************************************************************************')
```

I observe that:

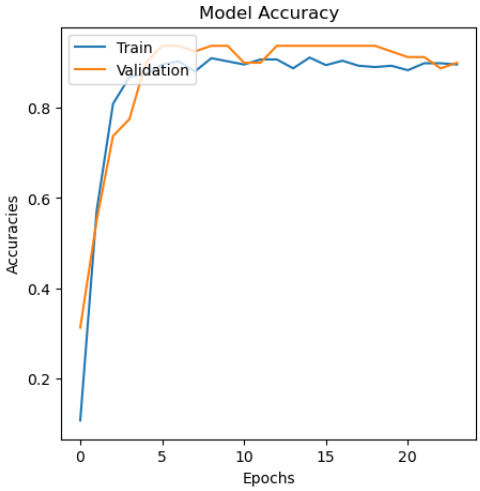
Hidden Size	Loss	Accuracy	Val Loss	Val Acc	Test Loss	Test Acc
265	1.3121	0.8919	0.2878	0.9125	4.2667	0.4898
512	1.2574	0.8904	0.2633	0.9375	4.2490	0.5354
1024	0.8917	0.9045	0.2029	0.9250	4.2094	0.5505

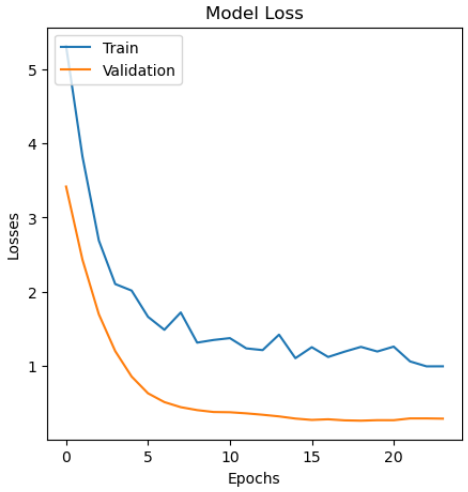
1. Hidden size = 256



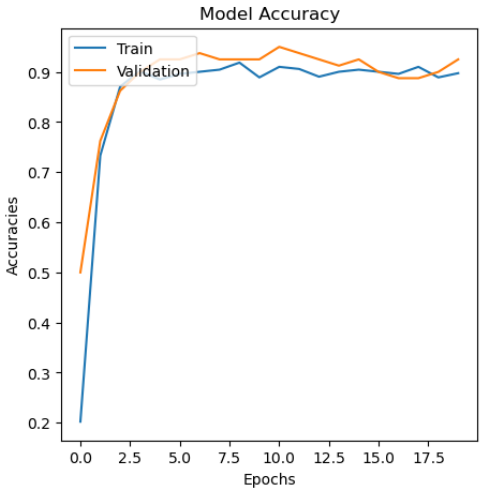


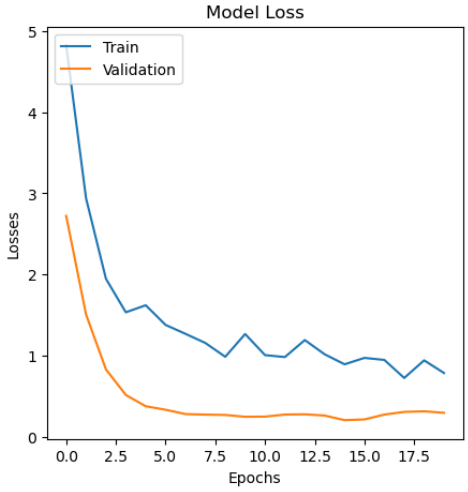
2. Hidden size = 512





3. Hidden size = 1024





3- Dropout Trails:

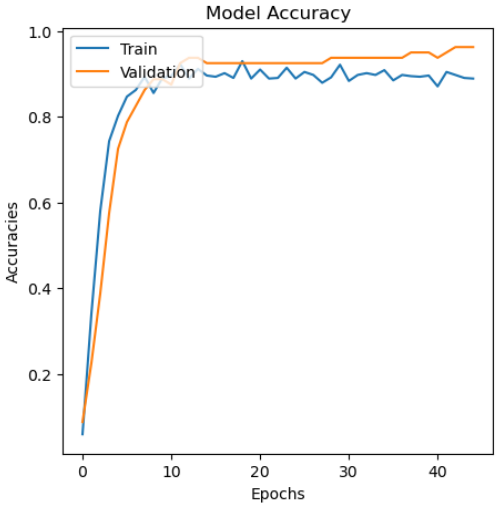
I tired three values [0.1, 0.3, 0.5]:

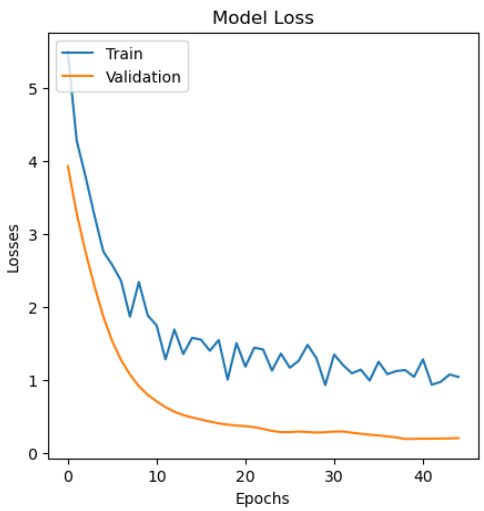
```
dropout_rates = [0.1, 0.3, 0.5]
print(f'Start Dropout Rate Trails of values: {dropout_rates}')
for dropout_rate in dropout_rates:
    print('**********************************************************************************************')
    print(f'Start of Dropout Rate: {dropout_rate}, Trail')
    print()
    model = create_model_Adam(hidden = 256, with_dropout = True, dropout_rate = dropout_rate, lr = 0.001, l2 = None)
    print()
    history = model.fit(X_train, y_train, batch_size = 128, epochs = 50, validation_split = 0.1, callbacks=[callback])
    print()
    display_accuracy(history)
    print()
    display_val_accuracy(history)
    print()
    evaluate(model)
    print()
    print(f'End of Trail')
    print('**********************************************************************************************')
```

I observe that:

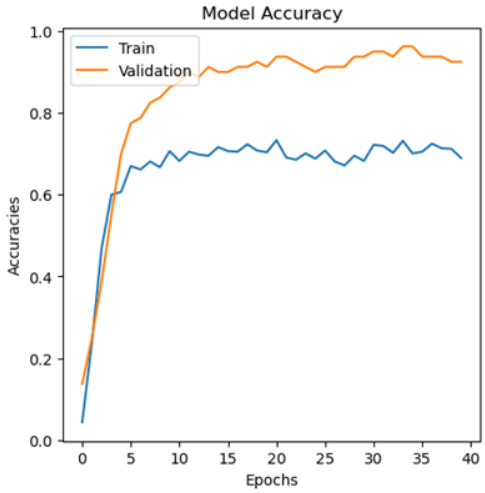
Dropout Rate	Loss	Accuracy	Val Loss	Val Acc	Test Loss	Test Acc
0.1	1.0465	0.8961	0.1968	0.9500	4.1880	0.4293
0.3	2.3997	0.7008	0.1774	0.9625	4.0535	0.5404
0.5	2.4291	0.5506	0.1924	0.9500	3.8850	0.4697

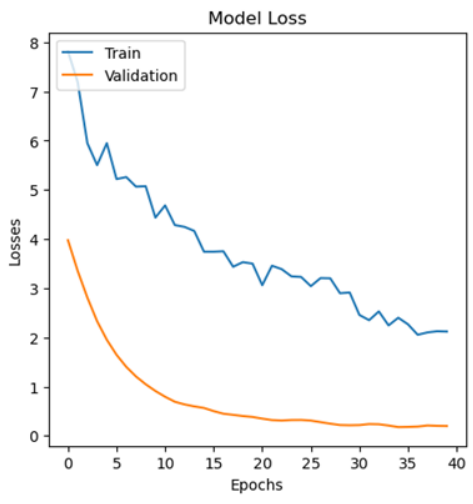
1. Dropout Rate = 0.1



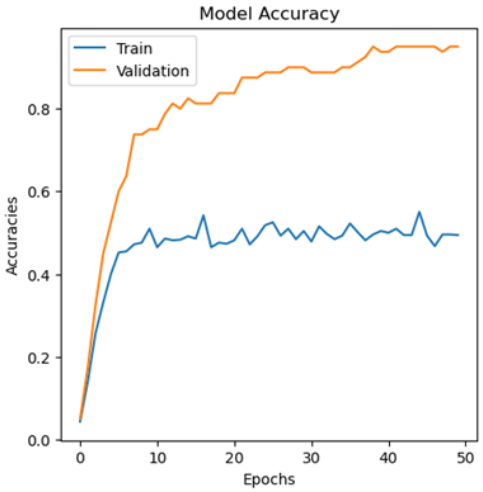


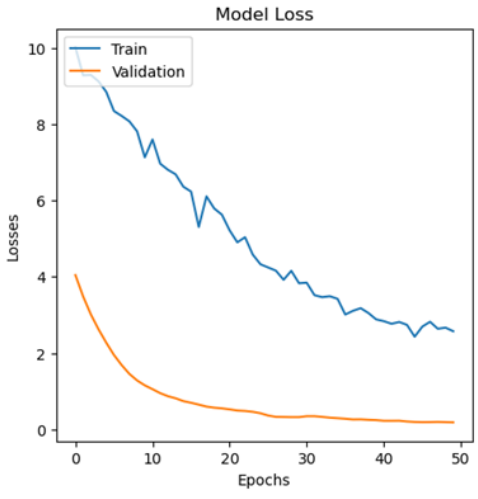
2. Dropout Rate = 0.3





3. Dropout Rate = 0.5





4- Learning Rates Trails:

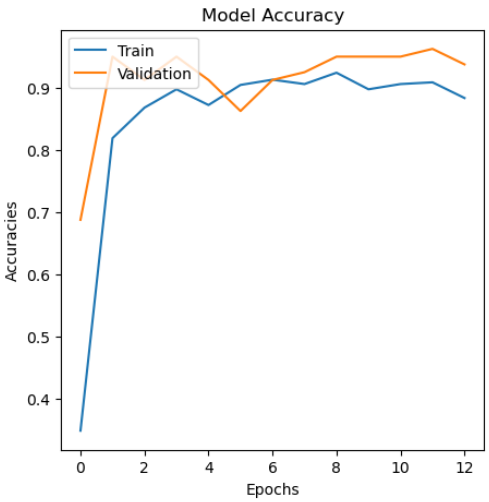
I tired three values [0.01, 0.001, 0.0001]:

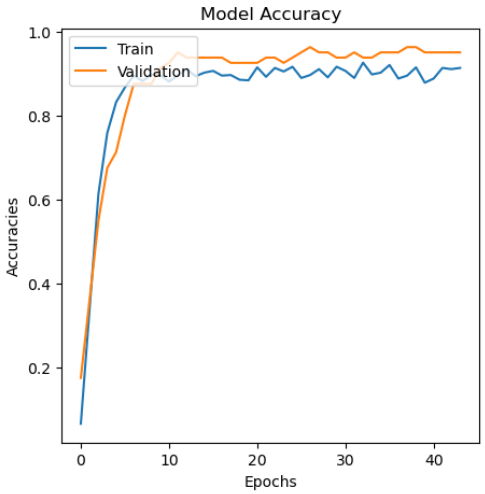
```
learning_rates = [0.01, 0.001, 0.0001]
print(f'Start Learning Rate Trails of values: {learning_rates}')
for learning_rate in learning_rates:
    print('**********************************************************************************************')
    print(f'Start of Learning Rate: {learning_rate}, Trail')
    print()
    model = create_model_Adam(hidden = 256, with_dropout = True, dropout_rate = 0.1, lr = learning_rate, l2 = None)
    print()
    history = model.fit(X_train, y_train, batch_size = 128, epochs = 50, validation_split = 0.1, callbacks=[callback])
    print()
    display_accuracy(history)
    print()
    display_val_accuracy(history)
    print()
    evaluate(model)
    print()
    print(f'End of Trail')
    print('**********************************************************************************************')
```

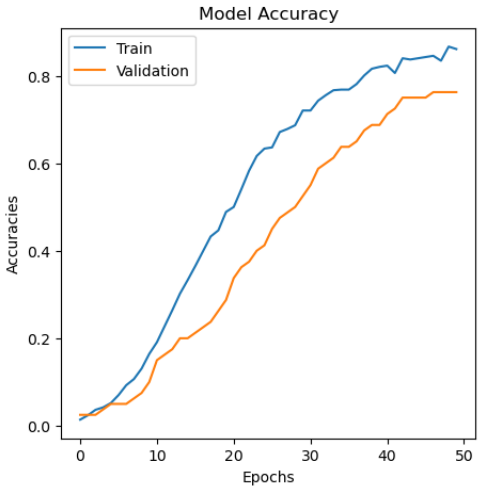
I observe that:

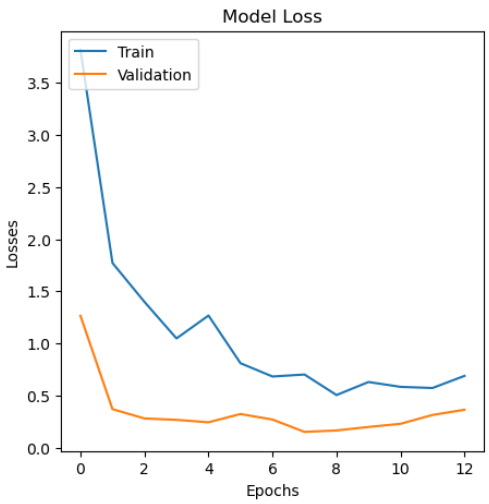
Learning Rate	Loss	Accuracy	Val Loss	Val Acc	Test Loss	Test Acc
0.01	0.7017	0.9059	0.1512	0.9250	3.5691	0.4242
0.001	0.9059	0.9143	0.1781	0.9625	4.1981	0.5051
0.0001	2.6986	0.8610	1.8842	0.7625	4.4934	0.4242

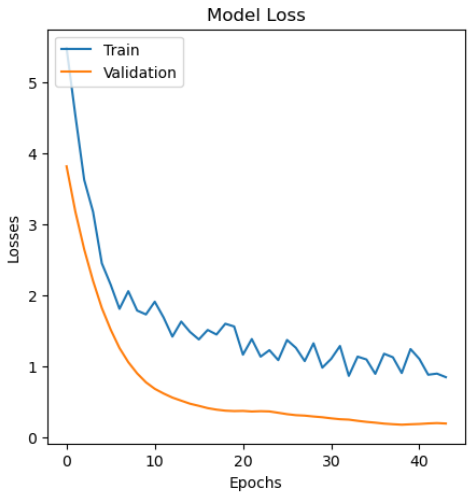
1. Learning Rate = 0.01
2. Learning Rate = 0.001
3. Learning Rate = 0.0001

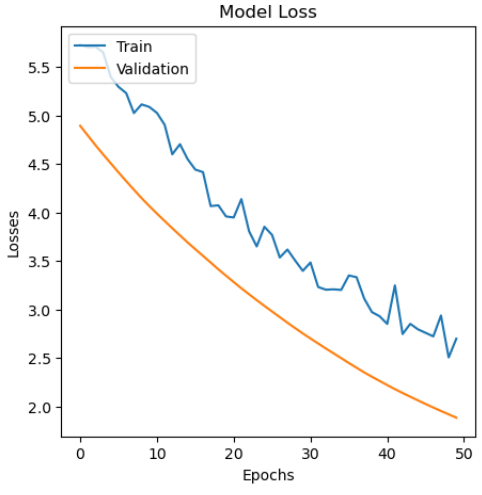












I also Tried other hyperparameters in the Notebook like:

- Regularization (Weight Decay - L2) values: [0.01, 0.001, 0.0001]
  - The best value was **0.001** with Test Accuracy = **0.5404**
- Different Optimizers (SGD - RMSP)
  - SGD was the worst optimizer of the three I had tested here [Adam, SGD, RMSP] it reaches only **2%** of Test Accuracy.
  - RMSP was better than the SGD and it was near to Adam in performance, it reaches **44%** of Test Accuracy.

The Main Notebook is attached under title **Project 1**. I build this report on it.

I also tried removing the features with high correlation above **0.9** which reduce the number of features from **192** to **129**, that make a good effect on the test accuracy on all the previous trails.

And Reached **75%** Test Accuracy at Hidden Trail **1024**.

I made that in a separate notebook, I attached it also under title **Project 1 – Less Features**.

Notes:

I had implemented this Project on **Kaggle** Notebook that is attached with the competition Leaf Classification | Kaggle.

Thank you.
Adham Mokhtar
20398545