Leaf Disease Classification - Keras CNN .. With 96% Accuracy

In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
import matplotlib.pyplot as plt
tf.compat.v1.set_random_seed(0)
from tensorflow import keras
import numpy as np
np.random.seed(0)
import itertools
from keras.preprocessing.image import image_dataset_from_directory
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from sklearn.metrics import precision_score, accuracy_score, recall_score, confusion_mat
```

Data Loading

Setting up Image Data Generators

In [2]:

```
#train_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/
#test_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/

train_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/
                                         image_size=(256, 256))
test_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/N
                                        image_size=(256, 256))

rescale = Rescaling(scale=1.0/255)
train_gen = train_gen.map(lambda image,label:(rescale(image),label))
test_gen  = test_gen.map(lambda image,label:(rescale(image),label))
```

```
Found 70295 files belonging to 38 classes.

2022-06-02 09:49:51.830329: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:51.990947: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:51.991830: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:51.994026: I tensorflow/core/platform/cpu_feature_guard.c
c:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-crit
ical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriat
e compiler flags.
2022-06-02 09:49:51.994343: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:51.995141: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:51.995803: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:54.246926: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:54.247814: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:54.248494: I tensorflow/stream_executor/cuda/cuda_gpu_exe
cutor.cc:937] successful NUMA node read from SysFS had negative value (-
1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 09:49:54.249073: I tensorflow/core/common_runtime/gpu/gpu_devic
e.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 wit
h 15403 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:00:04.0, compute capability: 6.0

Found 17572 files belonging to 38 classes.
```

Data Engineering

Since the data is already augmented, there is no requirement of data engineering. Feature scaling is automatically done by image generators

Modelling

In [3]:

```
model = keras.Sequential()

model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same",input_shape=(256
model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))

model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(1568,activation="relu"))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(38,activation="softmax"))

opt = keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=opt,loss="sparse_categorical_crossentropy",metrics=['accuracy'])
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
conv2d_1 (Conv2D)	(None, 256, 256, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 85, 85, 32)	0
conv2d_2 (Conv2D)	(None, 85, 85, 64)	18496
conv2d_3 (Conv2D)	(None, 85, 85, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_4 (Conv2D)	(None, 28, 28, 128)	73856
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 9, 9, 128)	0
conv2d_6 (Conv2D)	(None, 9, 9, 256)	295168
conv2d_7 (Conv2D)	(None, 9, 9, 256)	590080
conv2d_8 (Conv2D)	(None, 9, 9, 512)	3277312
conv2d_9 (Conv2D)	(None, 9, 9, 512)	6554112
flatten (Flatten)	(None, 41472)	0
dense (Dense)	(None, 1568)	65029664
dropout (Dropout)	(None, 1568)	0
dense_1 (Dense)	(None, 38)	59622

Total params: 76,092,966
Trainable params: 76,092,966
Non-trainable params: 0

In [4]:

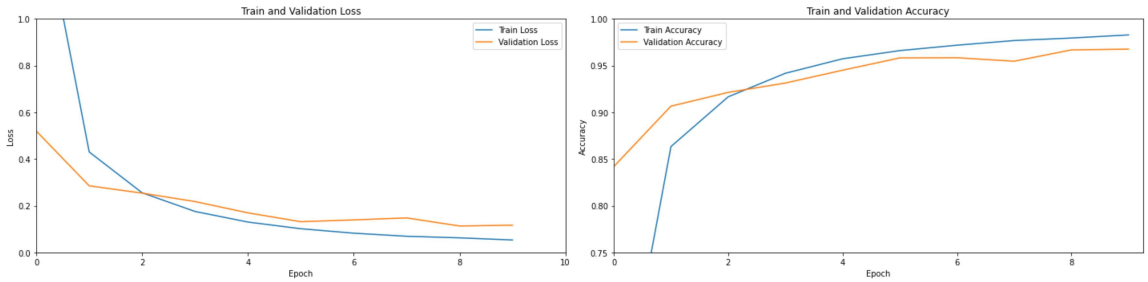
```
ep = 10
history = model.fit_generator(train_gen,
          validation_data=test_gen,
          epochs = ep)
```

```
Epoch 1/10

2022-06-02 09:49:57.725110: I tensorflow/compiler/mlir/mlir_graph_optimiza
tion_pass.cc:185] None of the MLIR Optimization Passes are enabled (regist
ered 2)
2022-06-02 09:50:00.285178: I tensorflow/stream_executor/cuda/cuda_dnn.cc:
369] Loaded cuDNN version 8005

2197/2197 [==============================] - 316s 140ms/step - loss: 1.602
2 - accuracy: 0.5339 - val_loss: 0.5197 - val_accuracy: 0.8419
Epoch 2/10
2197/2197 [==============================] - 195s 89ms/step - loss: 0.4302
- accuracy: 0.8633 - val_loss: 0.2852 - val_accuracy: 0.9066
Epoch 3/10
2197/2197 [==============================] - 194s 88ms/step - loss: 0.2553
- accuracy: 0.9166 - val_loss: 0.2539 - val_accuracy: 0.9214
Epoch 4/10
2197/2197 [==============================] - 195s 88ms/step - loss: 0.1754
- accuracy: 0.9418 - val_loss: 0.2177 - val_accuracy: 0.9314
Epoch 5/10
2197/2197 [==============================] - 193s 88ms/step - loss: 0.1299
- accuracy: 0.9573 - val_loss: 0.1694 - val_accuracy: 0.9451
Epoch 6/10
2197/2197 [==============================] - 194s 88ms/step - loss: 0.1014
- accuracy: 0.9661 - val_loss: 0.1315 - val_accuracy: 0.9583
Epoch 7/10
2197/2197 [==============================] - 194s 88ms/step - loss: 0.0823
- accuracy: 0.9719 - val_loss: 0.1392 - val_accuracy: 0.9585
Epoch 8/10
2197/2197 [==============================] - 204s 93ms/step - loss: 0.0691
- accuracy: 0.9769 - val_loss: 0.1477 - val_accuracy: 0.9547
Epoch 9/10
2197/2197 [==============================] - 194s 88ms/step - loss: 0.0623
- accuracy: 0.9796 - val_loss: 0.1131 - val_accuracy: 0.9668
Epoch 10/10
2197/2197 [==============================] - 194s 88ms/step - loss: 0.0532
- accuracy: 0.9829 - val_loss: 0.1166 - val_accuracy: 0.9677
```

Metrics

In [5]:

```
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
plt.title("Train and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.plot(history.history['loss'],label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.xlim(0, 10)
plt.ylim(0.0,1.0)
plt.legend()

plt.subplot(1,2,2)
plt.title("Train and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.xlim(0, 9.25)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()
```



In [6]:

```
labels = []
predictions = []
for x,y in test_gen:
    labels.append(list(y.numpy()))
    predictions.append(tf.argmax(model.predict(x),1).numpy())
```

In [7]:

```
predictions = list(itertools.chain.from_iterable(predictions))
labels = list(itertools.chain.from_iterable(labels))
```

In [8]:

```
print("Train Accuracy  : {:.2f} %".format(history.history['accuracy'][-1]*100))
print("Test Accuracy   : {:.2f} %".format(accuracy_score(labels, predictions) * 100))
print("Precision Score : {:.2f} %".format(precision_score(labels, predictions, average='
print("Recall Score    : {:.2f} %".format(recall_score(labels, predictions, average='mic
```

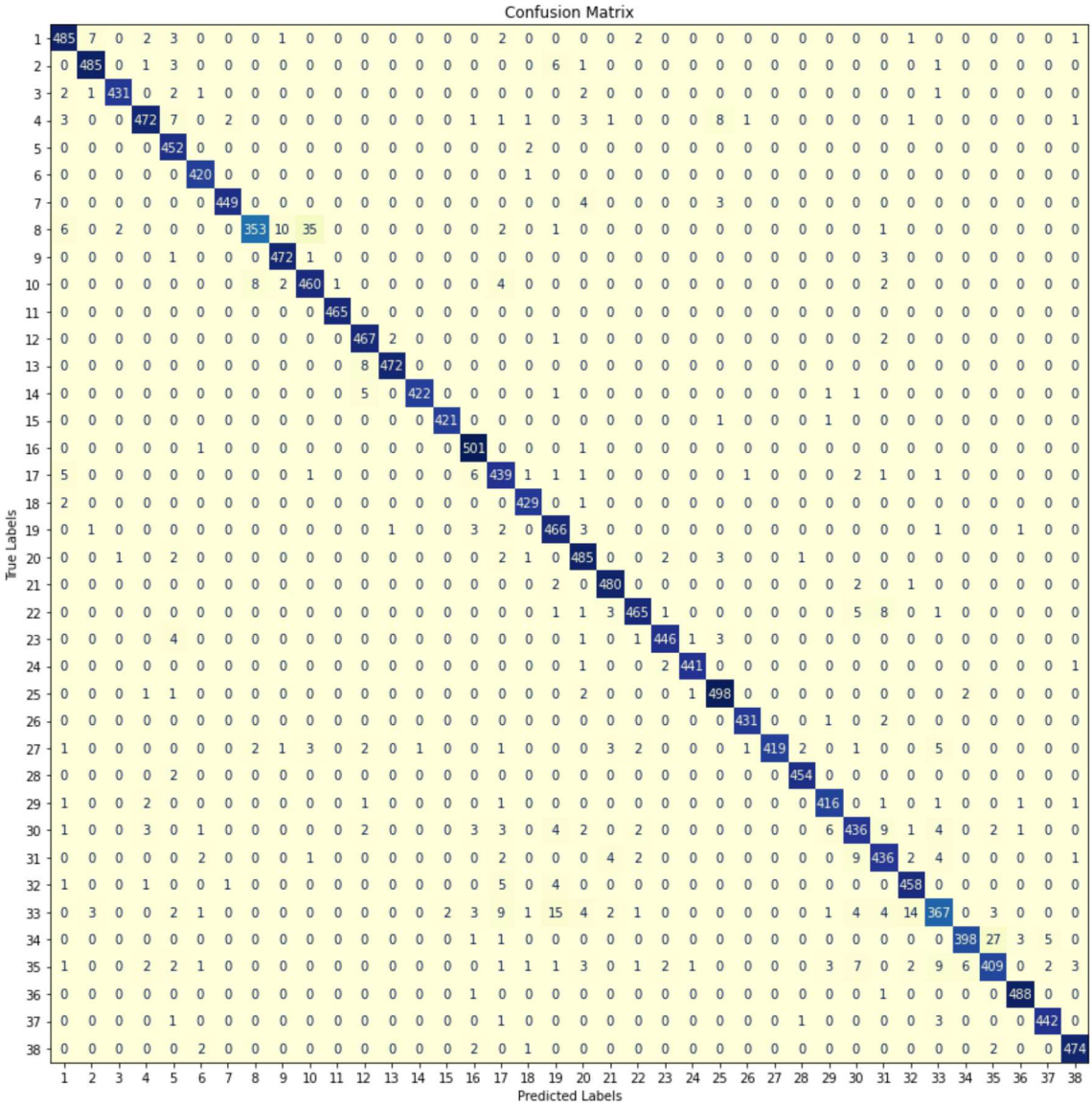
```
Train Accuracy  : 98.29 %
Test Accuracy   : 96.77 %
Precision Score : 96.77 %
Recall Score    : 96.77 %
```

Confusion Matrix

In [9]:

```
plt.figure(figsize= (20,5))
cm = confusion_matrix(labels, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=list(range(1,39)))
fig, ax = plt.subplots(figsize=(15,15))
disp.plot(ax=ax,colorbar= False,cmap = 'YlGnBu')
plt.title("Confusion Matrix")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

```
<Figure size 1440x360 with 0 Axes>
```



In [ ]:

```
```