

Introduction

In this notebook, I've used **CNN** to perform Image Classification on the Nail Disease dataset. Since this dataset is small, if we train a neural network to it, it won't really give us a good result. Therefore, I'm going to use the concept of **Transfer Learning** to train the model to get accurate results. Please install matplotlib, numpy, seaborn, tensorflow, opencv, and pandas.

Importing Libraries

```
In [1]: import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow.keras import layers
from tqdm import tqdm
import os
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, TensorBoard,
from sklearn.metrics import classification_report, confusion_matrix
import ipywidgets as widgets
import io
from PIL import Image
from IPython.display import display, clear_output
from warnings import filterwarnings
for dirname, _, filenames in os.walk(r'C:\Ken\Documents\Py\ModelTrain\NailDiseases'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

localhost:8888/nbconvert/html/FullCNNNail.ipynb?download=false

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

22/57

[illegible]

24/57

25/57

26/57

[illegible]

[illegible]

29/57

[illegible]

[illegible]

32/57

[illegible]

```

C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (55).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (56).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (57).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (58).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (59).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (60).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (61).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (62).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (63).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (64).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (65).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (66).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (67).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (68).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (69).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (70).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (71).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (72).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (73).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (74).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (75).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (76).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (77).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (78).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (79).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (80).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (81).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (82).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (83).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (84).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (85).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (86).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (87).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (88).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (89).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (90).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (91).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (92).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (93).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (94).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (95).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (96).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (97).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (98).PNG
C:\Ken\Documents\Py\ModelTrain\NailDiseases\Training\Yellow Nail\1 (99).PNG

```

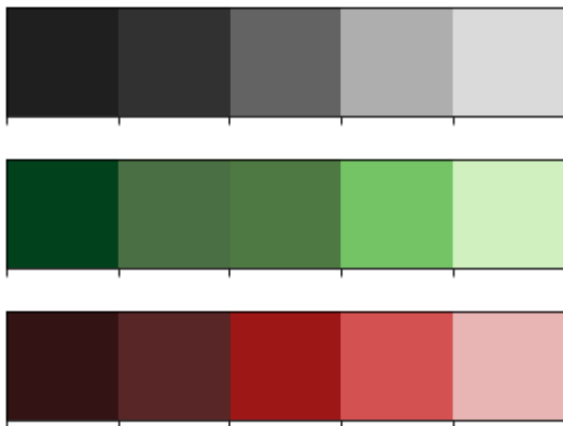
Color

```

In [2]: #Colors for graphs
colors_dark = ["#1F1F1F", "#313131", "#636363", "#AEAEAE", "#DADADA"]
colors_red = ["#331313", "#582626", "#9E1717", "#D35151", "#E9B4B4"]
colors_green = ["#01411C", "#4B6F44", "#4F7942", "#74C365", "#D0F0C0"]

sns.palplot(colors_dark)
sns.palplot(colors_green)
sns.palplot(colors_red)

```



Data Preparation

```
In [3]: labels = ['Alopecia Areata', 'Clubbing', 'Muehrckes Lines',
                  'Splinter Hemorrhage', 'White Nail', 'Yellow Nail']
```

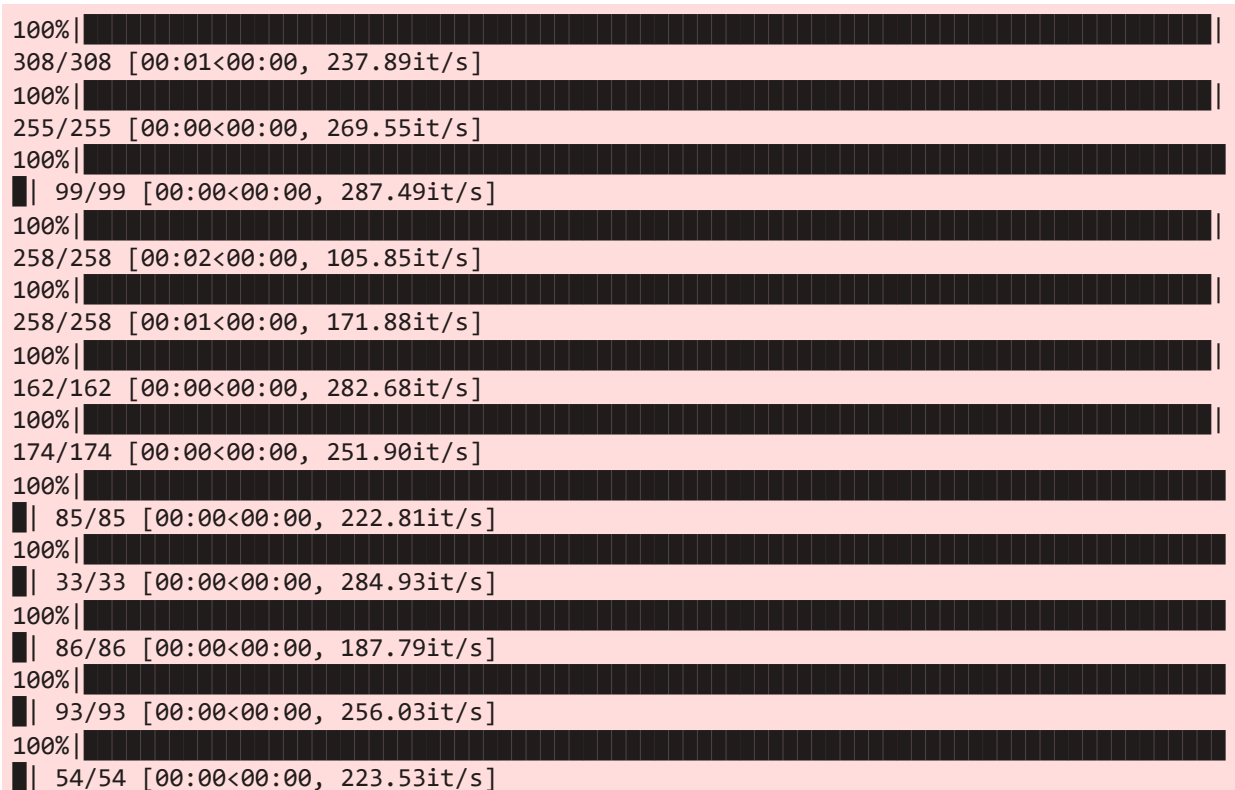
We start off by appending all the images from the directories into a Python list and then converting them into numpy arrays after resizing it.

```
In [4]: X_train = []
        y_train = []
        image_size = 224

        for i in labels:
            folderPath = os.path.join(r'C:\Ken\Documents\Py\ModelTrain\NailDiseases', 'Training')
            for j in tqdm(os.listdir(folderPath)):
                img = cv2.imread(os.path.join(folderPath, j))
                img = cv2.resize(img, (image_size, image_size))
                RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                X_train.append(RGB_img)
                y_train.append(i)

        for i in labels:
            folderPath = os.path.join(r'C:\Ken\Documents\Py\ModelTrain\NailDiseases', 'Testing')
            for j in tqdm(os.listdir(folderPath)):
                img = cv2.imread(os.path.join(folderPath, j))
                img = cv2.resize(img, (image_size, image_size))
                RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                X_train.append(RGB_img)
                y_train.append(i)

        X_train = np.array(X_train)
        y_train = np.array(y_train)
```



```
In [5]: k=0

fig, ax = plt.subplots(1,4,figsize=(25,25))
fig.text(s='Sample Image From Each Label',size=18,fontweight='bold',
        fontname='monospace',color=colors_dark[1],y=0.62,x=0.4,alpha=0.8)
try:
    for i in labels:
        j=0
        while True :
            if y_train[j]==i:
                ax[k].imshow(X_train[j])
                ax[k].set_title(y_train[j])
                ax[k].axis('off')
                k+=1
                break
            j+=1
except IndexError as e:
    pass
```

Out[8]: (1865, 224, 224, 3)

Dividing the dataset into **Training** and **Testing** sets.

```
In [9]: X_train,X_test,y_train,y_test = train_test_split(X_train,y_train, test_size=0.1,random
```

Performing **One Hot Encoding** on the labels after converting it into numerical values:

```
In [10]: y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)

y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

Transfer Learning

Deep convolutional neural network models may take days or even weeks to train on very large datasets.

In this notebook, I'll be using the **EfficientNetB0** model which will use the weights from the **ImageNet** dataset.

The `include_top` parameter is set to *False* so that the network doesn't include the top layer/output layer from the pre-built model which allows us to add our own output layer depending upon our use case.

```
In [11]: data_augmentation = tf.keras.Sequential(
[
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
]
)
```

```
In [12]: effnet = EfficientNetB0(weights='imagenet',include_top=False,input_shape=(image_size,i
```

```
In [13]: model = data_augmentation
model = effnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dense(6,activation='softmax')(model)
model = tf.keras.models.Model(inputs=effnet.input, outputs = model)
```

```
In [14]: model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
=====			
input_1 (InputLayer)	[(None, 224, 224, 3) 0]		[]
rescaling (Rescaling)	(None, 224, 224, 3) 0		['input_1[0][0]']
normalization (Normalization)	(None, 224, 224, 3) 7		['rescaling[0][0]']
tf.math.truediv (TFOpLambda)	(None, 224, 224, 3) 0		['normalization[0][0]']
stem_conv_pad (ZeroPadding2D)	(None, 225, 225, 3) 0		['tf.math.truediv[0][0]']
stem_conv (Conv2D)	(None, 112, 112, 32) 864		['stem_conv_pad[0][0]']
stem_bn (BatchNormalization)	(None, 112, 112, 32) 128		['stem_conv[0][0]']
stem_activation (Activation)	(None, 112, 112, 32) 0		['stem_bn[0][0]']
block1a_dwconv (DepthwiseConv2D)	(None, 112, 112, 32) 288		['stem_activation[0][0]']
block1a_bn (BatchNormalization)	(None, 112, 112, 32) 128		['block1a_dwconv[0][0]']
block1a_activation (Activation)	(None, 112, 112, 32) 0		['block1a_bn[0][0]']
block1a_se_squeeze (GlobalAveragePooling2D)	(None, 32)	0	['block1a_activation[0][0]']
block1a_se_reshape (Reshape)	(None, 1, 1, 32)	0	['block1a_se_squeeze[0][0]']
block1a_se_reduce (Conv2D)	(None, 1, 1, 8)	264	['block1a_se_reshape[0][0]']
block1a_se_expand (Conv2D)	(None, 1, 1, 32)	288	['block1a_se_reduce[0][0]']
block1a_se_excite (Multiply)	(None, 112, 112, 32) 0		['block1a_activation[0][0]', ['block1a_se_expand[0][0]']
block1a_project_conv (Conv2D)	(None, 112, 112, 16) 512		['block1a_se_excite[0][0]']

)			
block1a_project_bn (BatchNormalization)	(None, 112, 112, 16)	64	['block1a_project_co nv[0][0]']
)			
block2a_expand_conv (Conv2D)	(None, 112, 112, 96)	1536	['block1a_project_bn [0][0]']
)			
block2a_expand_bn (BatchNormalization)	(None, 112, 112, 96)	384	['block2a_expand_con v[0][0]']
)			
block2a_expand_activation (Activation)	(None, 112, 112, 96)	0	['block2a_expand_bn [0][0]']
)			
block2a_dwconv_pad (ZeroPadding2D)	(None, 113, 113, 96)	0	['block2a_expand_act ivation[0][0]']
)			
block2a_dwconv (DepthwiseConv2D)	(None, 56, 56, 96)	864	['block2a_dwconv_pad [0][0]']
)			
block2a_bn (BatchNormalization)	(None, 56, 56, 96)	384	['block2a_dwconv[0] [0]']
)			
block2a_activation (Activation)	(None, 56, 56, 96)	0	['block2a_bn[0][0]']
)			
block2a_se_squeeze (GlobalAveragePooling2D)	(None, 96)	0	['block2a_activation [0][0]']
)			
block2a_se_reshape (Reshape)	(None, 1, 1, 96)	0	['block2a_se_squeeze [0][0]']
)			
block2a_se_reduce (Conv2D)	(None, 1, 1, 4)	388	['block2a_se_reshape [0][0]']
)			
block2a_se_expand (Conv2D)	(None, 1, 1, 96)	480	['block2a_se_reduce [0][0]']
)			
block2a_se_excite (Multiply)	(None, 56, 56, 96)	0	['block2a_activation [0][0]', 'block2a_se_expand [0][0]']
)			
block2a_project_conv (Conv2D)	(None, 56, 56, 24)	2304	['block2a_se_excite [0][0]']
)			
block2a_project_bn (BatchNormalization)	(None, 56, 56, 24)	96	['block2a_project_co nv[0][0]']
)			
block2b_expand_conv (Conv2D)	(None, 56, 56, 144)	3456	['block2a_project_bn [0][0]']

block2b_expand_bn (BatchNormalization)	(None, 56, 56, 144)	576	['block2b_expand_con
v[0][0]']			
block2b_expand_activation (Activation)	(None, 56, 56, 144)	0	['block2b_expand_bn
[0][0]']			
block2b_dwconv (DepthwiseConv2D)	(None, 56, 56, 144)	1296	['block2b_expand_act
ivation[0][0]']			
block2b_bn (BatchNormalization)	(None, 56, 56, 144)	576	['block2b_dwconv[0]
[0]']			
block2b_activation (Activation)	(None, 56, 56, 144)	0	['block2b_bn[0][0]']
[0][0]']			
block2b_se_squeeze (GlobalAveragePooling2D)	(None, 144)	0	['block2b_activation
[0][0]']			
block2b_se_reshape (Reshape)	(None, 1, 1, 144)	0	['block2b_se_squeeze
[0][0]']			
block2b_se_reduce (Conv2D)	(None, 1, 1, 6)	870	['block2b_se_reshape
[0][0]']			
block2b_se_expand (Conv2D)	(None, 1, 1, 144)	1008	['block2b_se_reduce
[0][0]']			
block2b_se_excite (Multiply)	(None, 56, 56, 144)	0	['block2b_activation
[0][0]',			
[0][0]']			'block2b_se_expand
block2b_project_conv (Conv2D)	(None, 56, 56, 24)	3456	['block2b_se_excite
[0][0]']			
block2b_project_bn (BatchNormalization)	(None, 56, 56, 24)	96	['block2b_project_co
nv[0][0]']			
block2b_drop (Dropout)	(None, 56, 56, 24)	0	['block2b_project_bn
[0][0]']			
block2b_add (Add)	(None, 56, 56, 24)	0	['block2b_drop[0]
[0]',			
[0][0]']			'block2a_project_bn
block3a_expand_conv (Conv2D)	(None, 56, 56, 144)	3456	['block2b_add[0]
[0]']			
block3a_expand_bn (BatchNormalization)	(None, 56, 56, 144)	576	['block3a_expand_con
v[0][0]']			
ization)			

block3a_expand_activation (Activation[0][0])	(None, 56, 56, 144)	0	['block3a_expand_bn[0][0]']
block3a_dwconv_pad (DepthwiseConv2D)	(None, 59, 59, 144)	0	['block3a_expand_activation[0][0]']
block3a_dwconv (DepthwiseConv2D)	(None, 28, 28, 144)	3600	['block3a_dwconv_pad[0][0]']
block3a_bn (BatchNormalization)	(None, 28, 28, 144)	576	['block3a_dwconv[0][0]']
block3a_activation (Activation)	(None, 28, 28, 144)	0	['block3a_bn[0][0]']
block3a_se_squeeze (GlobalAveragePooling2D)	(None, 144)	0	['block3a_activation[0][0]']
block3a_se_reshape (Reshape)	(None, 1, 1, 144)	0	['block3a_se_squeeze[0][0]']
block3a_se_reduce (Conv2D)	(None, 1, 1, 6)	870	['block3a_se_reshape[0][0]']
block3a_se_expand (Conv2D)	(None, 1, 1, 144)	1008	['block3a_se_reduce[0][0]']
block3a_se_excite (Multiply)	(None, 28, 28, 144)	0	['block3a_activation[0][0]', 'block3a_se_expand[0][0]']
block3a_project_conv (Conv2D)	(None, 28, 28, 40)	5760	['block3a_se_excite[0][0]']
block3a_project_bn (BatchNormalization)	(None, 28, 28, 40)	160	['block3a_project_conv[0][0]']
block3b_expand_conv (Conv2D)	(None, 28, 28, 240)	9600	['block3a_project_bn[0][0]']
block3b_expand_bn (BatchNormalization)	(None, 28, 28, 240)	960	['block3b_expand_conv[0][0]']
block3b_expand_activation (Activation)	(None, 28, 28, 240)	0	['block3b_expand_bn[0][0]']
block3b_dwconv (DepthwiseConv2D)	(None, 28, 28, 240)	6000	['block3b_expand_activation[0][0]']
block3b_bn (BatchNormalization)	(None, 28, 28, 240)	960	['block3b_dwconv[0][0]']

```

[0]']
)

block3b_activation (Activation (None, 28, 28, 240) 0 ['block3b_bn[0][0]']
)

block3b_se_squeeze (GlobalAveragePooling2D) (None, 240) 0 ['block3b_activation
[0][0]']

block3b_se_reshape (Reshape) (None, 1, 1, 240) 0 ['block3b_se_squeeze
[0][0]']

block3b_se_reduce (Conv2D) (None, 1, 1, 10) 2410 ['block3b_se_reshape
[0][0]']

block3b_se_expand (Conv2D) (None, 1, 1, 240) 2640 ['block3b_se_reduce
[0][0]']

block3b_se_excite (Multiply) (None, 28, 28, 240) 0 ['block3b_activation
[0][0]',
'block3b_se_expand
[0][0]']

block3b_project_conv (Conv2D) (None, 28, 28, 40) 9600 ['block3b_se_excite
[0][0]']

block3b_project_bn (BatchNormalization) (None, 28, 28, 40) 160 ['block3b_project_co
nv[0][0]']

block3b_drop (Dropout) (None, 28, 28, 40) 0 ['block3b_project_bn
[0][0]']

block3b_add (Add) (None, 28, 28, 40) 0 ['block3b_drop[0]
[0]',
'block3a_project_bn
[0][0]']

block4a_expand_conv (Conv2D) (None, 28, 28, 240) 9600 ['block3b_add[0]
[0]']

block4a_expand_bn (BatchNormalization) (None, 28, 28, 240) 960 ['block4a_expand_con
v[0][0]']

block4a_expand_activation (Activation) (None, 28, 28, 240) 0 ['block4a_expand_bn
[0][0]']

block4a_dwconv_pad (ZeroPadding2D) (None, 29, 29, 240) 0 ['block4a_expand_act
ivation[0][0]']

block4a_dwconv (DepthwiseConv2D) (None, 14, 14, 240) 2160 ['block4a_dwconv_pad
[0][0]']

block4a_bn (BatchNormalization) (None, 14, 14, 240) 960 ['block4a_dwconv[0]
[0]']

```

)				
block4a_activation (Activation)	(None, 14, 14, 240)	0		['block4a_bn[0][0]']
)				
block4a_se_squeeze (GlobalAveragePooling2D)	(None, 240)	0		['block4a_activation[0][0]']
block4a_se_reshape (Reshape)	(None, 1, 1, 240)	0		['block4a_se_squeeze[0][0]']
block4a_se_reduce (Conv2D)	(None, 1, 1, 10)	2410		['block4a_se_reshape[0][0]']
block4a_se_expand (Conv2D)	(None, 1, 1, 240)	2640		['block4a_se_reduce[0][0]']
block4a_se_excite (Multiply)	(None, 14, 14, 240)	0		['block4a_activation[0][0]', 'block4a_se_expand[0][0]']
block4a_project_conv (Conv2D)	(None, 14, 14, 80)	19200		['block4a_se_excite[0][0]']
block4a_project_bn (BatchNormalization)	(None, 14, 14, 80)	320		['block4a_project_conv[0][0]']
block4b_expand_conv (Conv2D)	(None, 14, 14, 480)	38400		['block4a_project_bn[0][0]']
block4b_expand_bn (BatchNormalization)	(None, 14, 14, 480)	1920		['block4b_expand_conv[0][0]']
block4b_expand_activation (Activation)	(None, 14, 14, 480)	0		['block4b_expand_bn[0][0]']
block4b_dwconv (DepthwiseConv2D)	(None, 14, 14, 480)	4320		['block4b_expand_activation[0][0]']
block4b_bn (BatchNormalization)	(None, 14, 14, 480)	1920		['block4b_dwconv[0][0]']
)				
block4b_activation (Activation)	(None, 14, 14, 480)	0		['block4b_bn[0][0]']
)				
block4b_se_squeeze (GlobalAveragePooling2D)	(None, 480)	0		['block4b_activation[0][0]']
block4b_se_reshape (Reshape)	(None, 1, 1, 480)	0		['block4b_se_squeeze[0][0]']
block4b_se_reduce (Conv2D)	(None, 1, 1, 20)	9620		['block4b_se_reshape[0][0]']

[0][0]'				
block4b_se_expand (Conv2D)	(None, 1, 1, 480)	10080	['block4b_se_reduce	
[0][0]'				
block4b_se_excite (Multiply)	(None, 14, 14, 480)	0	['block4b_activation	
[0][0]',				
[0][0]'			'block4b_se_expand	
block4b_project_conv (Conv2D)	(None, 14, 14, 80)	38400	['block4b_se_excite	
[0][0]'				
block4b_project_bn (BatchNormal	(None, 14, 14, 80)	320	['block4b_project_co	
nv[0][0]'				
lization)				
block4b_drop (Dropout)	(None, 14, 14, 80)	0	['block4b_project_bn	
[0][0]'				
block4b_add (Add)	(None, 14, 14, 80)	0	['block4b_drop[0]	
[0]',				
[0][0]'			'block4a_project_bn	
block4c_expand_conv (Conv2D)	(None, 14, 14, 480)	38400	['block4b_add[0]	
[0]'				
block4c_expand_bn (BatchNormal	(None, 14, 14, 480)	1920	['block4c_expand_con	
v[0][0]'				
lization)				
block4c_expand_activation (Act	(None, 14, 14, 480)	0	['block4c_expand_bn	
[0][0]'				
ivation)				
block4c_dwconv (DepthwiseConv2	(None, 14, 14, 480)	4320	['block4c_expand_act	
ivation[0][0]				
D)			']	
block4c_bn (BatchNormalization	(None, 14, 14, 480)	1920	['block4c_dwconv[0]	
[0]'				
)				
block4c_activation (Activation	(None, 14, 14, 480)	0	['block4c_bn[0][0]'	
)				
block4c_se_squeeze (GlobalAver	(None, 480)	0	['block4c_activation	
[0][0]'				
agePooling2D)				
block4c_se_reshape (Reshape)	(None, 1, 1, 480)	0	['block4c_se_squeeze	
[0][0]'				
block4c_se_reduce (Conv2D)	(None, 1, 1, 20)	9620	['block4c_se_reshape	
[0][0]'				
block4c_se_expand (Conv2D)	(None, 1, 1, 480)	10080	['block4c_se_reduce	
[0][0]'				

block4c_se_excite (Multiply) [0][0]',	(None, 14, 14, 480)	0	['block4c_activation [0][0]', 'block4c_se_expand [0][0]']
block4c_project_conv (Conv2D) [0][0]']	(None, 14, 14, 80)	38400	['block4c_se_excite [0][0]']
block4c_project_bn (BatchNormal ization) nv[0][0]']	(None, 14, 14, 80)	320	['block4c_project_co nv[0][0]']
block4c_drop (Dropout) [0][0]']	(None, 14, 14, 80)	0	['block4c_project_bn [0][0]']
block4c_add (Add) [0]', [0]']	(None, 14, 14, 80)	0	['block4c_drop[0] 'block4b_add[0] [0]']
block5a_expand_conv (Conv2D) [0]']	(None, 14, 14, 480)	38400	['block4c_add[0] [0]']
block5a_expand_bn (BatchNormal ization) v[0][0]']	(None, 14, 14, 480)	1920	['block5a_expand_con v[0][0]']
block5a_expand_activation (Act ivation) [0][0]']	(None, 14, 14, 480)	0	['block5a_expand_bn [0][0]']
block5a_dwconv (DepthwiseConv2 D) ivation[0][0]	(None, 14, 14, 480)	12000	['block5a_expand_act ivation[0][0]
block5a_bn (BatchNormalization [0]'])	(None, 14, 14, 480)	1920	['block5a_dwconv[0] [0]']
block5a_activation (Activation)	(None, 14, 14, 480)	0	['block5a_bn[0][0]']
block5a_se_squeeze (GlobalAver agePooling2D) [0][0]']	(None, 480)	0	['block5a_activation [0][0]']
block5a_se_reshape (Reshape) [0][0]']	(None, 1, 1, 480)	0	['block5a_se_squeeze [0][0]']
block5a_se_reduce (Conv2D) [0][0]']	(None, 1, 1, 20)	9620	['block5a_se_reshape [0][0]']
block5a_se_expand (Conv2D) [0][0]']	(None, 1, 1, 480)	10080	['block5a_se_reduce [0][0]']
block5a_se_excite (Multiply) [0][0]', [0][0]']	(None, 14, 14, 480)	0	['block5a_activation 'block5a_se_expand [0][0]']

block5a_project_conv (Conv2D)	(None, 14, 14, 112)	53760	['block5a_se_excite [0][0]']
block5a_project_bn (BatchNormal ization)	(None, 14, 14, 112)	448	['block5a_project_co nv[0][0]']
block5b_expand_conv (Conv2D)	(None, 14, 14, 672)	75264	['block5a_project_bn [0][0]']
block5b_expand_bn (BatchNormal ization)	(None, 14, 14, 672)	2688	['block5b_expand_con v[0][0]']
block5b_expand_activation (Act ivation)	(None, 14, 14, 672)	0	['block5b_expand_bn [0][0]']
block5b_dwconv (DepthwiseConv2 D)	(None, 14, 14, 672)	16800	['block5b_expand_act ivation[0][0]']
block5b_bn (BatchNormalization)	(None, 14, 14, 672)	2688	['block5b_dwconv[0] [0]']
block5b_activation (Activation)	(None, 14, 14, 672)	0	['block5b_bn[0][0]']
block5b_se_squeeze (GlobalAver agePooling2D)	(None, 672)	0	['block5b_activation [0][0]']
block5b_se_reshape (Reshape)	(None, 1, 1, 672)	0	['block5b_se_squeeze [0][0]']
block5b_se_reduce (Conv2D)	(None, 1, 1, 28)	18844	['block5b_se_reshape [0][0]']
block5b_se_expand (Conv2D)	(None, 1, 1, 672)	19488	['block5b_se_reduce [0][0]']
block5b_se_excite (Multiply)	(None, 14, 14, 672)	0	['block5b_activation [0][0]', 'block5b_se_expand [0][0]']
block5b_project_conv (Conv2D)	(None, 14, 14, 112)	75264	['block5b_se_excite [0][0]']
block5b_project_bn (BatchNormal ization)	(None, 14, 14, 112)	448	['block5b_project_co nv[0][0]']
block5b_drop (Dropout)	(None, 14, 14, 112)	0	['block5b_project_bn [0][0]']
block5b_add (Add)	(None, 14, 14, 112)	0	['block5b_drop[0] [0]', 'block5a_project_bn

[0][0]']

block5c_expand_conv (Conv2D) (None, 14, 14, 672) 75264
[0]']

['block5b_add[0]

block5c_expand_bn (BatchNormal
v[0][0]')
ization)

['block5c_expand_con

block5c_expand_activation (Act (None, 14, 14, 672) 0
[0][0]')
ivation)

['block5c_expand_bn

block5c_dwconv (DepthwiseConv2 (None, 14, 14, 672) 16800
ivation[0][0]
D)

['block5c_expand_act
']

block5c_bn (BatchNormalization (None, 14, 14, 672) 2688
[0]')
)

['block5c_dwconv[0]

block5c_activation (Activation (None, 14, 14, 672) 0
)

['block5c_bn[0][0]']

block5c_se_squeeze (GlobalAver (None, 672) 0
[0][0]')
agePooling2D)

['block5c_activation

block5c_se_reshape (Reshape) (None, 1, 1, 672) 0
[0][0]']

['block5c_se_squeeze

block5c_se_reduce (Conv2D) (None, 1, 1, 28) 18844
[0][0]']

['block5c_se_reshape

block5c_se_expand (Conv2D) (None, 1, 1, 672) 19488
[0][0]']

['block5c_se_reduce

block5c_se_excite (Multiply) (None, 14, 14, 672) 0
[0][0]',

['block5c_activation

[0][0]']

'block5c_se_expand

block5c_project_conv (Conv2D) (None, 14, 14, 112) 75264
[0][0]']

['block5c_se_excite

block5c_project_bn (BatchNorma (None, 14, 14, 112) 448
nv[0][0]')
lization)

['block5c_project_co

block5c_drop (Dropout) (None, 14, 14, 112) 0
[0][0]']

['block5c_project_bn

block5c_add (Add) (None, 14, 14, 112) 0
[0]',

['block5c_drop[0]

[0]']

'block5b_add[0]

block6a_expand_conv (Conv2D) (None, 14, 14, 672) 75264
[0]']

['block5c_add[0]

block6a_expand_bn (BatchNormalization)	(None, 14, 14, 672)	2688	['block6a_expand_con v[0][0]']
block6a_expand_activation (Activation)	(None, 14, 14, 672)	0	['block6a_expand_bn [0][0]']
block6a_dwconv_pad (ZeroPadding2D)	(None, 17, 17, 672)	0	['block6a_expand_act ivation[0][0]']
block6a_dwconv (DepthwiseConv2D)	(None, 7, 7, 672)	16800	['block6a_dwconv_pad [0][0]']
block6a_bn (BatchNormalization)	(None, 7, 7, 672)	2688	['block6a_dwconv[0] [0]']
block6a_activation (Activation)	(None, 7, 7, 672)	0	['block6a_bn[0][0]']
block6a_se_squeeze (GlobalAveragePooling2D)	(None, 672)	0	['block6a_activation [0][0]']
block6a_se_reshape (Reshape)	(None, 1, 1, 672)	0	['block6a_se_squeeze [0][0]']
block6a_se_reduce (Conv2D)	(None, 1, 1, 28)	18844	['block6a_se_reshape [0][0]']
block6a_se_expand (Conv2D)	(None, 1, 1, 672)	19488	['block6a_se_reduce [0][0]']
block6a_se_excite (Multiply)	(None, 7, 7, 672)	0	['block6a_activation [0][0]', 'block6a_se_expand [0][0]']
block6a_project_conv (Conv2D)	(None, 7, 7, 192)	129024	['block6a_se_excite [0][0]']
block6a_project_bn (BatchNormalization)	(None, 7, 7, 192)	768	['block6a_project_co nv[0][0]']
block6b_expand_conv (Conv2D)	(None, 7, 7, 1152)	221184	['block6a_project_bn [0][0]']
block6b_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block6b_expand_con v[0][0]']
block6b_expand_activation (Activation)	(None, 7, 7, 1152)	0	['block6b_expand_bn [0][0]']
block6b_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28800	['block6b_expand_act


```

ivation[0][0]
D)

block6b_bn (BatchNormalization (None, 7, 7, 1152) 4608 ['block6b_dwconv[0]
[0]']
)

block6b_activation (Activation (None, 7, 7, 1152) 0 ['block6b_bn[0][0]']
)

block6b_se_squeeze (GlobalAveragePooling2D) (None, 1152) 0 ['block6b_activation
[0][0]']

block6b_se_reshape (Reshape) (None, 1, 1, 1152) 0 ['block6b_se_squeeze
[0][0]']

block6b_se_reduce (Conv2D) (None, 1, 1, 48) 55344 ['block6b_se_reshape
[0][0]']

block6b_se_expand (Conv2D) (None, 1, 1, 1152) 56448 ['block6b_se_reduce
[0][0]']

block6b_se_excite (Multiply) (None, 7, 7, 1152) 0 ['block6b_activation
[0][0]',
'block6b_se_expand
[0][0]']

block6b_project_conv (Conv2D) (None, 7, 7, 192) 221184 ['block6b_se_excite
[0][0]']

block6b_project_bn (BatchNormalization) (None, 7, 7, 192) 768 ['block6b_project_co
nv[0][0]']

block6b_drop (Dropout) (None, 7, 7, 192) 0 ['block6b_project_bn
[0][0]']

block6b_add (Add) (None, 7, 7, 192) 0 ['block6b_drop[0]
[0]',
'block6a_project_bn
[0][0]']

block6c_expand_conv (Conv2D) (None, 7, 7, 1152) 221184 ['block6b_add[0]
[0]']

block6c_expand_bn (BatchNormalization) (None, 7, 7, 1152) 4608 ['block6c_expand_con
v[0][0]']

block6c_expand_activation (Activation) (None, 7, 7, 1152) 0 ['block6c_expand_bn
[0][0]']

block6c_dwconv (DepthwiseConv2D) (None, 7, 7, 1152) 28800 ['block6c_expand_act
ivation[0][0]']

block6c_bn (BatchNormalization) (None, 7, 7, 1152) 4608 ['block6c_dwconv[0]
[0]']

```

```

)

block6c_activation (Activation (None, 7, 7, 1152) 0 ['block6c_bn[0][0]']
)

block6c_se_squeeze (GlobalAveragePooling2D) (None, 1152) 0 ['block6c_activation
[0][0]']

block6c_se_reshape (Reshape) (None, 1, 1, 1152) 0 ['block6c_se_squeeze
[0][0]']

block6c_se_reduce (Conv2D) (None, 1, 1, 48) 55344 ['block6c_se_reshape
[0][0]']

block6c_se_expand (Conv2D) (None, 1, 1, 1152) 56448 ['block6c_se_reduce
[0][0]']

block6c_se_excite (Multiply) (None, 7, 7, 1152) 0 ['block6c_activation
[0][0]',
'block6c_se_expand
[0][0]']

block6c_project_conv (Conv2D) (None, 7, 7, 192) 221184 ['block6c_se_excite
[0][0]']

block6c_project_bn (BatchNormalization) (None, 7, 7, 192) 768 ['block6c_project_co
nv[0][0]']

block6c_drop (Dropout) (None, 7, 7, 192) 0 ['block6c_project_bn
[0][0]']

block6c_add (Add) (None, 7, 7, 192) 0 ['block6c_drop[0]
[0]',
'block6b_add[0]
[0]']

block6d_expand_conv (Conv2D) (None, 7, 7, 1152) 221184 ['block6c_add[0]
[0]']

block6d_expand_bn (BatchNormalization) (None, 7, 7, 1152) 4608 ['block6d_expand_con
v[0][0]']

block6d_expand_activation (Activation) (None, 7, 7, 1152) 0 ['block6d_expand_bn
[0][0]']

block6d_dwconv (DepthwiseConv2D) (None, 7, 7, 1152) 28800 ['block6d_expand_act
ivation[0][0]']

block6d_bn (BatchNormalization) (None, 7, 7, 1152) 4608 ['block6d_dwconv[0]
[0]']

block6d_activation (Activation) (None, 7, 7, 1152) 0 ['block6d_bn[0][0]']
)

```

block6d_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	['block6d_activation[0][0]']
block6d_se_reshape (Reshape)	(None, 1, 1, 1152)	0	['block6d_se_squeeze[0][0]']
block6d_se_reduce (Conv2D)	(None, 1, 1, 48)	55344	['block6d_se_reshape[0][0]']
block6d_se_expand (Conv2D)	(None, 1, 1, 1152)	56448	['block6d_se_reduce[0][0]']
block6d_se_excite (Multiply)	(None, 7, 7, 1152)	0	['block6d_activation[0][0]', 'block6d_se_expand[0][0]']
block6d_project_conv (Conv2D)	(None, 7, 7, 192)	221184	['block6d_se_excite[0][0]']
block6d_project_bn (BatchNormalization)	(None, 7, 7, 192)	768	['block6d_project_conv[0][0]']
block6d_drop (Dropout)	(None, 7, 7, 192)	0	['block6d_project_bn[0][0]']
block6d_add (Add)	(None, 7, 7, 192)	0	['block6d_drop[0][0]', 'block6c_add[0][0]']
block7a_expand_conv (Conv2D)	(None, 7, 7, 1152)	221184	['block6d_add[0][0]']
block7a_expand_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block7a_expand_conv[0][0]']
block7a_expand_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_expand_bn[0][0]']
block7a_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	10368	['block7a_expand_activation[0][0]']
block7a_bn (BatchNormalization)	(None, 7, 7, 1152)	4608	['block7a_dwconv[0][0]']
block7a_activation (Activation)	(None, 7, 7, 1152)	0	['block7a_bn[0][0]']
block7a_se_squeeze (GlobalAveragePooling2D)	(None, 1152)	0	['block7a_activation[0][0]']
block7a_se_reshape (Reshape)	(None, 1, 1, 1152)	0	['block7a_se_squeeze[0][0]']

Training The Model

Note: The training takes 40 mins

```
In [17]: history = model.fit(X_train,y_train,validation_split=0.1, epochs = 7,verbose=1,batch_s
```

Epoch 1/7
69/69 [=====] - ETA: 0s - loss: 0.5447 - accuracy: 0.8185
Epoch 1: val_accuracy improved from -inf to 0.71429, saving model to effnet.h5
69/69 [=====] - 373s 5s/step - loss: 0.5447 - accuracy: 0.8185 - val_loss: 1.3688 - val_accuracy: 0.7143 - lr: 0.0010
Epoch 2/7
69/69 [=====] - ETA: 0s - loss: 0.1250 - accuracy: 0.9642
Epoch 2: val_accuracy improved from 0.71429 to 0.84524, saving model to effnet.h5
69/69 [=====] - 347s 5s/step - loss: 0.1250 - accuracy: 0.9642 - val_loss: 0.6283 - val_accuracy: 0.8452 - lr: 0.0010
Epoch 3/7
69/69 [=====] - ETA: 0s - loss: 0.1000 - accuracy: 0.9682
Epoch 3: val_accuracy improved from 0.84524 to 0.97024, saving model to effnet.h5
69/69 [=====] - 324s 5s/step - loss: 0.1000 - accuracy: 0.9682 - val_loss: 0.0468 - val_accuracy: 0.9702 - lr: 0.0010
Epoch 4/7
69/69 [=====] - ETA: 0s - loss: 0.0825 - accuracy: 0.9735
Epoch 4: val_accuracy did not improve from 0.97024
69/69 [=====] - 326s 5s/step - loss: 0.0825 - accuracy: 0.9735 - val_loss: 0.1977 - val_accuracy: 0.9345 - lr: 0.0010
Epoch 5/7
69/69 [=====] - ETA: 0s - loss: 0.0821 - accuracy: 0.9735
Epoch 5: val_accuracy did not improve from 0.97024
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
69/69 [=====] - 326s 5s/step - loss: 0.0821 - accuracy: 0.9735 - val_loss: 0.2845 - val_accuracy: 0.9345 - lr: 0.0010
Epoch 6/7
69/69 [=====] - ETA: 0s - loss: 0.0375 - accuracy: 0.9914
Epoch 6: val_accuracy improved from 0.97024 to 0.98214, saving model to effnet.h5
69/69 [=====] - 332s 5s/step - loss: 0.0375 - accuracy: 0.9914 - val_loss: 0.0345 - val_accuracy: 0.9821 - lr: 3.0000e-04
Epoch 7/7
69/69 [=====] - ETA: 0s - loss: 0.0169 - accuracy: 0.9954
Epoch 7: val_accuracy improved from 0.98214 to 1.00000, saving model to effnet.h5
69/69 [=====] - 330s 5s/step - loss: 0.0169 - accuracy: 0.9954 - val_loss: 0.0035 - val_accuracy: 1.0000 - lr: 3.0000e-04

```
In [18]: filterwarnings('ignore')

epochs = [i for i in range(7)]
fig, ax = plt.subplots(1,2,figsize=(10,6))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig.text(s='Epochs vs. Training and Validation Accuracy/Loss',size=12,fontweight='bold',
        fontname='monospace',color=colors_dark[1],y=1,x=0.28,alpha=0.8)

sns.despine()
ax[0].plot(epochs, train_acc, marker='o',markerfacecolor=colors_green[2],color=colors_
```

```

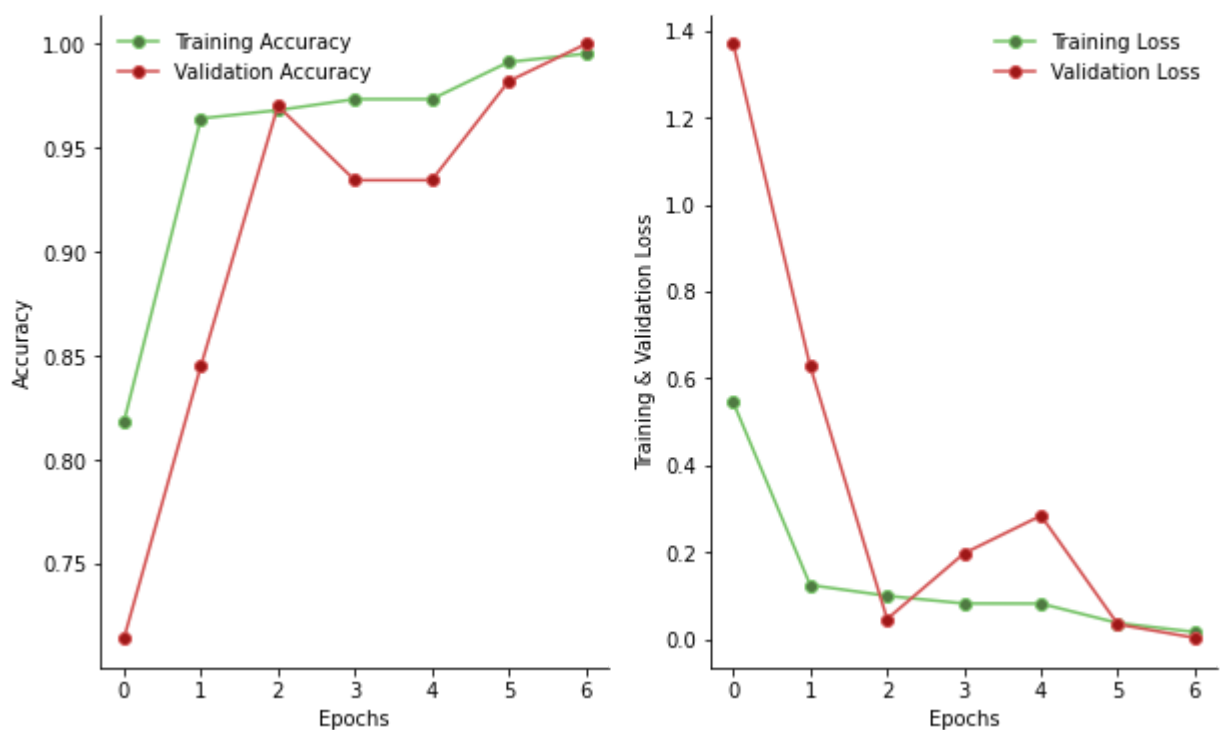
        label = 'Training Accuracy')
ax[0].plot(epochs, val_acc, marker='o',markerfacecolor=colors_red[2],color=colors_red[2],
        label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss, marker='o',markerfacecolor=colors_green[2],color=colors_green[2],
        label = 'Training Loss')
ax[1].plot(epochs, val_loss, marker='o',markerfacecolor=colors_red[2],color=colors_red[2],
        label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')

fig.show()

```

Epochs vs. Training and Validation Accuracy/Loss



Prediction

```

In [19]: pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)

```

6/6 [=====] - 11s 2s/step

Evaluation

In this,

- 1 - Alopecia Areata
- 2 - Clubbing
- 3 - Muehrckes Lines
- 4 - Splinter Hemorrhage
- 5 - White Nail
- 6 - Yellow Nail

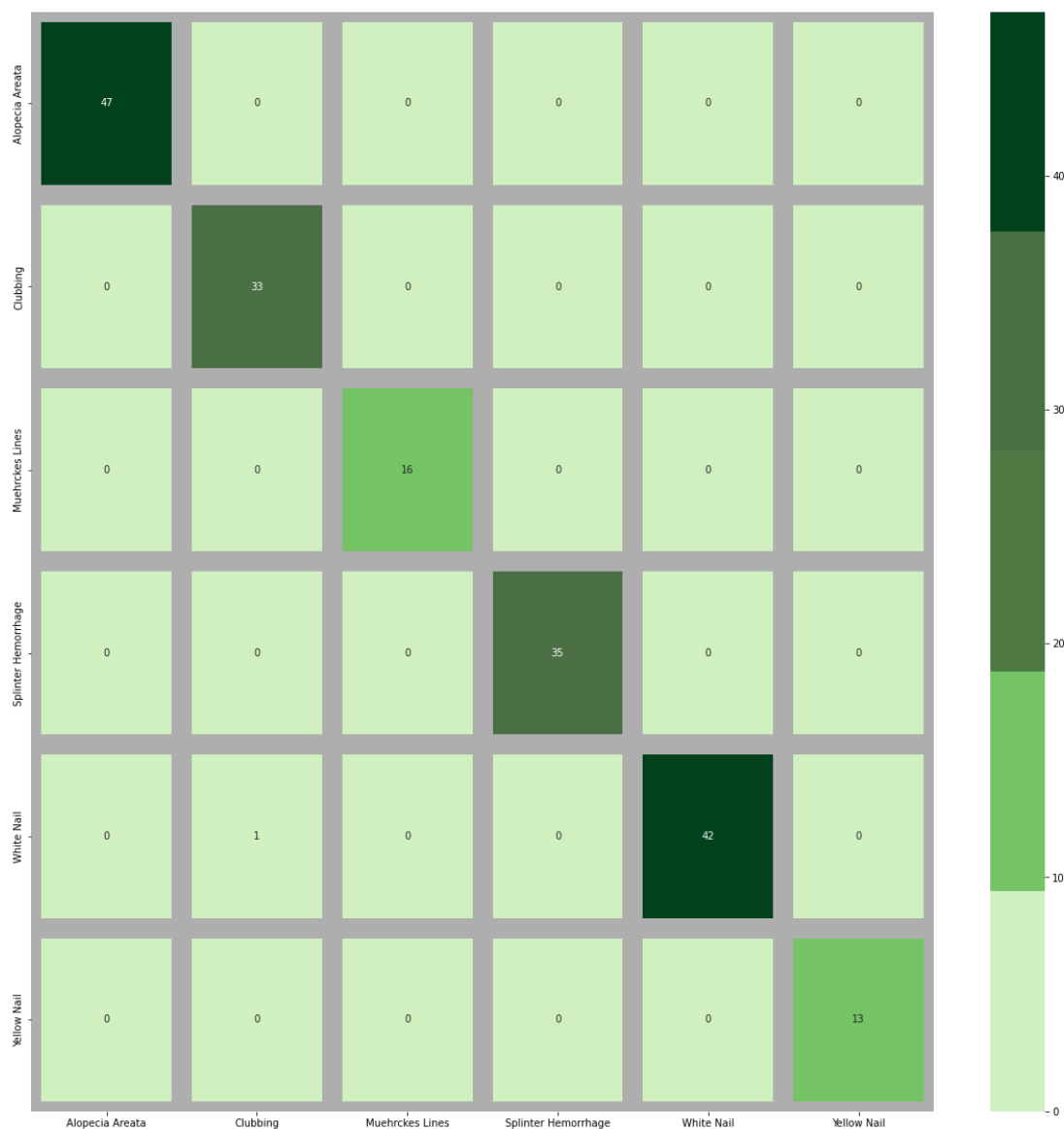
```
In [20]: print(classification_report(y_test_new,pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	47
1	0.97	1.00	0.99	33
2	1.00	1.00	1.00	16
3	1.00	1.00	1.00	35
4	1.00	0.98	0.99	43
5	1.00	1.00	1.00	13
accuracy			0.99	187
macro avg	1.00	1.00	1.00	187
weighted avg	0.99	0.99	0.99	187

```
In [21]: fig,ax=plt.subplots(1,1,figsize=(20,20))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels,yticklabels=labels,
            cmap=colors_green[:,::-1],alpha=1,linewidths=20,linecolor=colors_dark[3])
fig.text(s='Heatmap of the Confusion Matrix',size=50,fontweight='bold',
        fontname='monospace',color=colors_dark[1],y=0.92,x=0.28,alpha=0.8)

plt.show()
```

Heatmap of the Confusion Matrix



In [22]: `###Widget upload and predict`

```
In [24]: def img_pred(upload):
    for name, file_info in uploader.value.items():
        img = Image.open(io.BytesIO(file_info['content']))
        opencvImage = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)
        img = cv2.resize(opencvImage, (224, 224))
        img = img.reshape(1, 224, 224, 3)
        p = model.predict(img)
        p = np.argmax(p, axis=1)[0]

        if p==0:
            p='Alopecia Areata'
        elif p==1:
            p='Clubbing'
        elif p==2:
            p='Muehrckes Line'
```



```

elif p==3:
    p='Splinter Hemorrhage'
elif p==4:
    p='White Nail'
else:
    p='Yellow Nail'

if p!=0:
    print(f'The model predicts that it is a/an {p}')

```

This is where you can upload the image by clicking on the **Upload** button:

```

In [27]: uploader = widgets.FileUpload()
display(uploader)

FileUpload(value={}, description='Upload')

```

After uploading the image, you can click on the **Predict** button below to make predictions:

```

In [28]: button = widgets.Button(description='Predict')
out = widgets.Output()
def on_button_clicked(_):
    with out:
        clear_output()
        try:
            img_pred(uploader)

        except:
            print('No Image Uploaded/Invalid Image File')
button.on_click(on_button_clicked)
widgets.VBox([button,out])

VBox(children=(Button(description='Predict', style=ButtonStyle()), Output()))

```

In []: