Problem statement:

To build a CNN based model which can accurately detect melanoma. Melanoma is a type of cancer that can be deadly if not detected early. It accounts for 75% of skin cancer deaths. A solution which can evaluate images and alert the dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

Importing Skin Cancer Data

To do: Take necessary actions to read the data

Importing all the important libraries

In [1]:

```
import pathlib
import matplotlib.pyplot as plt

import numpy as np
import pandas as pd
import os
import PIL
from glob import glob
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import BatchNormalization
from tensorflow.python.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D,
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau
from keras.utils.np_utils import to_categorical
```

In [2]:

```
## If you are using the data by mounting the google drive, use the following :
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

This assignment uses a dataset of about 2357 images of skin cancer types. The dataset contains 9 sub-directories in each train and test subdirectories. The 9 sub-directories contains the images of 9 skin cancer types respectively.

In [3]:

```
# Defining the path for train and test images
root_path = '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data'
data_dir_train = pathlib.Path(root_path + '/Train')
data_dir_test = pathlib.Path(root_path + '/Test')
```

In [4]:

```
image_count_train = len(list(data_dir_train.glob('*/*.jpg')))
print(image_count_train)
image_count_test = len(list(data_dir_test.glob('*/*.jpg')))
print(image_count_test)
```

```
2239
118
```

Load using keras.preprocessing

Let's load these images off disk using the helpful image_dataset_from_directory utility.

Create a dataset

Define some parameters for the loader:

In [5]:

```
batch_size = 32
img_height = 180
img_width = 180

input_shape = (180,180,3)
num_classes=9
```

Use 80% of the images for training, and 20% for validation.

In [6]:

```
## Write your train dataset here
## Note use seed=123 while creating your dataset using tf.keras.preprocessing.image_dat
## Note, make sure your resize your images to the size img_height*img_width, while writ
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.2,
  labels='inferred',
  subset="training",
  label_mode='categorical',
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 2239 files belonging to 9 classes.
Using 1792 files for training.
```

In [7]:

```
## Write your validation dataset here
## Note use seed=123 while creating your dataset using tf.keras.preprocessing.image_dat
## Note, make sure your resize your images to the size img_height*img_width, while writ
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.2,
  labels='inferred',
  subset="validation",
  label_mode='categorical',
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 2239 files belonging to 9 classes.
Using 447 files for validation.
```

In [8]:

```
# List out all the classes of skin cancer and store them in a list.
# You can find the class names in the class_names attribute on these datasets.
# These correspond to the directory names in alphabetical order.
class_names = train_ds.class_names
print(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma',
'nevus', 'pigmented benign keratosis', 'seborrheic keratosis', 'squamous cel
l carcinoma', 'vascular lesion']
```

In [9]:

```
num_classes = len(class_names)
num_classes
```

Out[9]:

```
9
```

Visualize the data

Todo, create a code to visualize one instance of all the nine classes present in the dataset

In [10]:

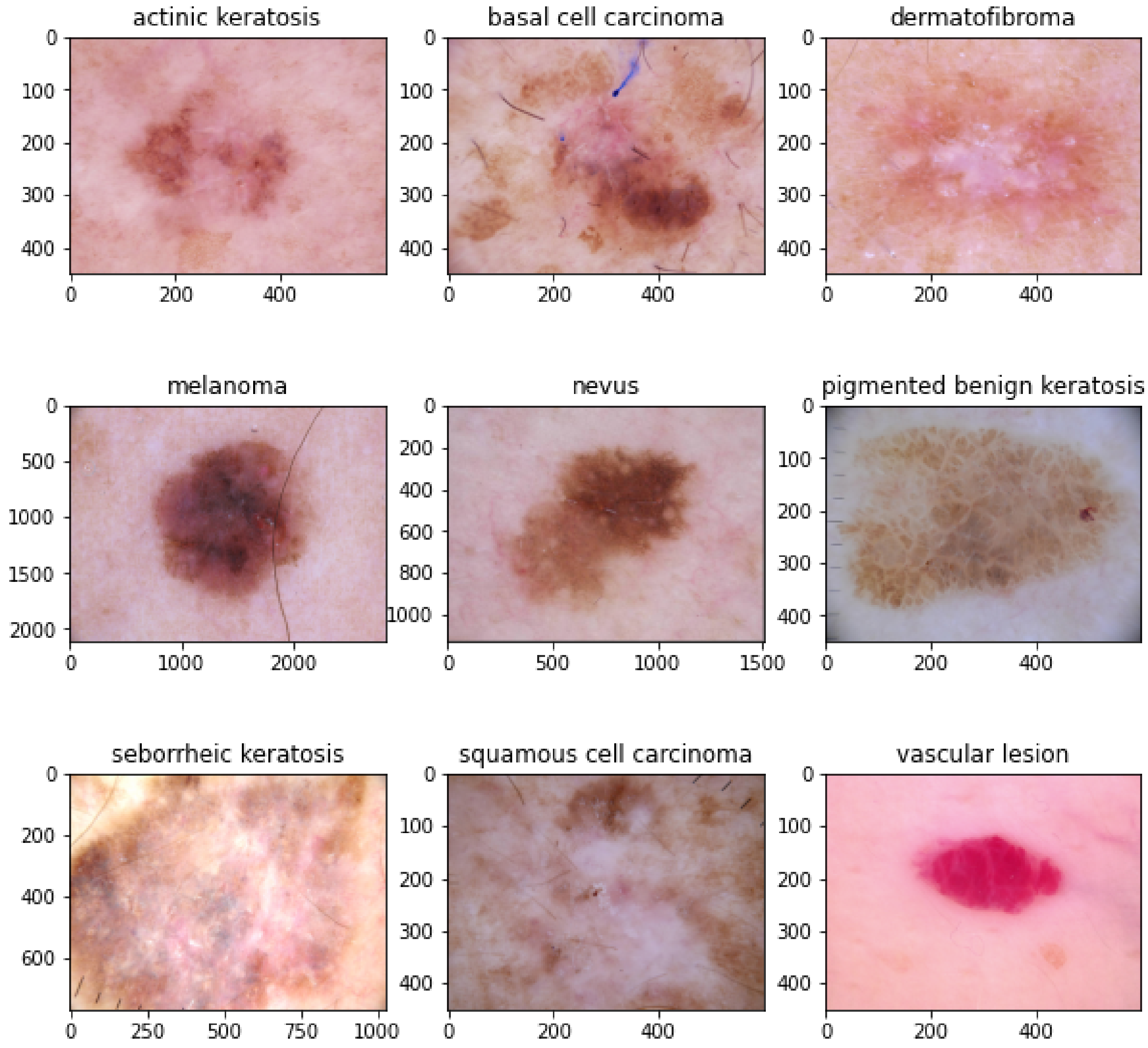
```
# A batch size of 2000 was taken as for train_ds the batch size was 32 and the batch mi
# Hence this additional step has been written.

all_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.1,
  labels='inferred',
  subset="training",
  label_mode='categorical',
  seed=123,
  image_size=(img_height, img_width),
  batch_size=2000)
```

```
Found 2239 files belonging to 9 classes.
Using 2016 files for training.
```

In [54]:

```
import matplotlib.image as mpimg
plt.figure(figsize=(10,10))
for i in range(9):
  plt.subplot(3, 3, i + 1)
  image = mpimg.imread(str(list(data_dir_train.glob(class_names[i]+'/*.jpg'))[1]))
  plt.title(class_names[i])
  plt.imshow(image)
```



The `image_batch` is a tensor of the shape `(32, 180, 180, 3)`. This is a batch of 32 images of shape `180x180x3` (the last dimension refers to color channels RGB). The `label_batch` is a tensor of the shape `(32,)`, these are corresponding labels to the 32 images.

`Dataset.cache()` keeps the images in memory after they're loaded off disk during the first epoch.

`Dataset.prefetch()` overlaps data preprocessing and model execution while training.

In [12]:

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Create the model

Todo: Create a CNN model, which can accurately detect 9 classes present in the dataset. Use `layers.experimental.preprocessing.Rescaling` to normalize pixel values between (0,1). The RGB channel values are in the `[0, 255]` range. This is not ideal for a neural network. Here, it is good to standardize values to be in the `[0, 1]`

In [13]:

```
model1 = Sequential()
model1.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(18

model1.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shap
model1.add(MaxPool2D(pool_size=(2, 2)))

model1.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2, 2)))

model1.add(Conv2D(128, kernel_size=(3, 3), padding='same', activation='relu'))
model1.add(MaxPool2D(pool_size=(2, 2)))

model1.add(Flatten())

model1.add(Dense(512, activation='relu'))
model1.add(Dropout(0.5))
model1.add(Dense(9, activation='softmax'))
```

Compile the model

Choose an appropirate optimiser and loss function for model training

In [14]:

```
model1.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [15]:

```
# View the summary of all layers
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 180, 180, 3)       0

 module_wrapper (ModuleWrapp  (None, 180, 180, 32)     896
 er)

 module_wrapper_1 (ModuleWra  (None, 90, 90, 32)       0
 pper)

 module_wrapper_2 (ModuleWra  (None, 90, 90, 64)       18496
 pper)

 module_wrapper_3 (ModuleWra  (None, 45, 45, 64)       0
 pper)

 module_wrapper_4 (ModuleWra  (None, 45, 45, 128)      73856
 pper)

 module_wrapper_5 (ModuleWra  (None, 22, 22, 128)      0
 pper)

 module_wrapper_6 (ModuleWra  (None, 61952)            0
 pper)

 module_wrapper_7 (ModuleWra  (None, 512)              31719936
 pper)

 module_wrapper_8 (ModuleWra  (None, 512)              0
 pper)

 module_wrapper_9 (ModuleWra  (None, 9)                4617
 pper)

=================================================================
Total params: 31,817,801
Trainable params: 31,817,801
Non-trainable params: 0
_________________________________________________________________
```

Train the model

In [16]:

```
epochs = 20
history = model1.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 23s 370ms/step - loss: 2.0018 - acc
uracy: 0.2958 - val_loss: 1.6474 - val_accuracy: 0.4094
Epoch 2/20
56/56 [==============================] - 18s 329ms/step - loss: 1.6142 - acc
uracy: 0.4074 - val_loss: 1.4828 - val_accuracy: 0.4787
Epoch 3/20
56/56 [==============================] - 18s 328ms/step - loss: 1.4607 - acc
uracy: 0.4877 - val_loss: 1.4717 - val_accuracy: 0.5347
Epoch 4/20
56/56 [==============================] - 18s 326ms/step - loss: 1.3819 - acc
uracy: 0.5201 - val_loss: 1.3723 - val_accuracy: 0.5369
Epoch 5/20
56/56 [==============================] - 18s 326ms/step - loss: 1.3045 - acc
uracy: 0.5463 - val_loss: 1.3891 - val_accuracy: 0.5213
Epoch 6/20
56/56 [==============================] - 18s 320ms/step - loss: 1.2640 - acc
uracy: 0.5519 - val_loss: 1.3574 - val_accuracy: 0.5078
Epoch 7/20
56/56 [==============================] - 18s 319ms/step - loss: 1.2293 - acc
uracy: 0.5742 - val_loss: 1.3281 - val_accuracy: 0.5615
Epoch 8/20
56/56 [==============================] - 18s 318ms/step - loss: 1.1301 - acc
uracy: 0.5960 - val_loss: 1.3616 - val_accuracy: 0.5324
Epoch 9/20
56/56 [==============================] - 18s 321ms/step - loss: 1.0912 - acc
uracy: 0.6150 - val_loss: 1.3217 - val_accuracy: 0.5481
Epoch 10/20
56/56 [==============================] - 18s 322ms/step - loss: 1.0570 - acc
uracy: 0.6233 - val_loss: 1.3029 - val_accuracy: 0.5660
Epoch 11/20
56/56 [==============================] - 18s 320ms/step - loss: 0.9615 - acc
uracy: 0.6507 - val_loss: 1.4093 - val_accuracy: 0.5280
Epoch 12/20
56/56 [==============================] - 18s 320ms/step - loss: 0.9233 - acc
uracy: 0.6574 - val_loss: 1.4278 - val_accuracy: 0.4989
Epoch 13/20
56/56 [==============================] - 18s 323ms/step - loss: 0.9745 - acc
uracy: 0.6479 - val_loss: 1.3529 - val_accuracy: 0.5570
Epoch 14/20
56/56 [==============================] - 18s 318ms/step - loss: 0.8536 - acc
uracy: 0.6942 - val_loss: 1.4878 - val_accuracy: 0.4944
Epoch 15/20
56/56 [==============================] - 18s 318ms/step - loss: 0.9203 - acc
uracy: 0.6853 - val_loss: 1.3254 - val_accuracy: 0.5772
Epoch 16/20
56/56 [==============================] - 18s 325ms/step - loss: 0.7456 - acc
uracy: 0.7282 - val_loss: 1.4653 - val_accuracy: 0.5369
Epoch 17/20
56/56 [==============================] - 18s 320ms/step - loss: 0.7642 - acc
uracy: 0.7148 - val_loss: 1.8352 - val_accuracy: 0.5302
Epoch 18/20
```

```
56/56 [==============================] - 18s 319ms/step - loss: 0.6970 - acc
uracy: 0.7433 - val_loss: 1.7941 - val_accuracy: 0.5101
Epoch 19/20
56/56 [==============================] - 18s 322ms/step - loss: 0.7462 - acc
uracy: 0.7232 - val_loss: 1.5955 - val_accuracy: 0.5459
Epoch 20/20
56/56 [==============================] - 18s 323ms/step - loss: 0.5914 - acc
uracy: 0.7801 - val_loss: 1.6580 - val_accuracy: 0.5615
```

Visualizing training results

In [17]:

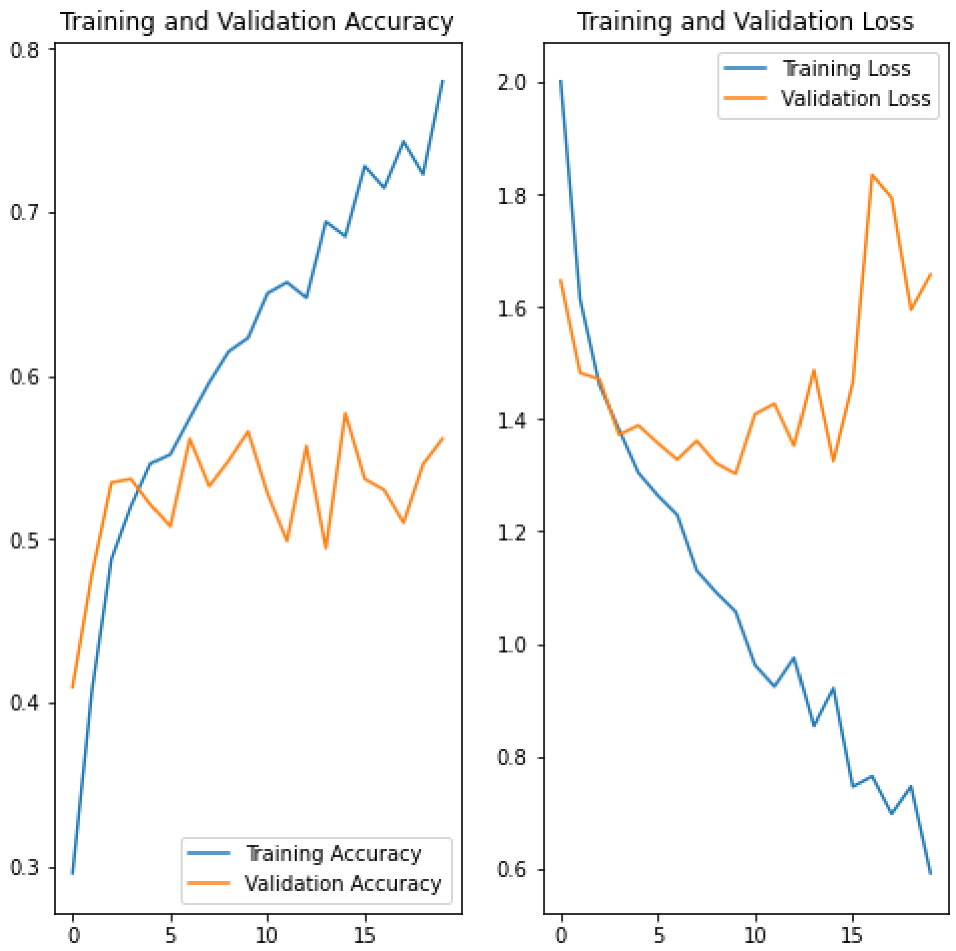
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Observation Model 1 - The model is overfitting as the training accuracy is high as 80% however validation accuracy is only 53%

Model 2 with Data Augmetation

In [18]:

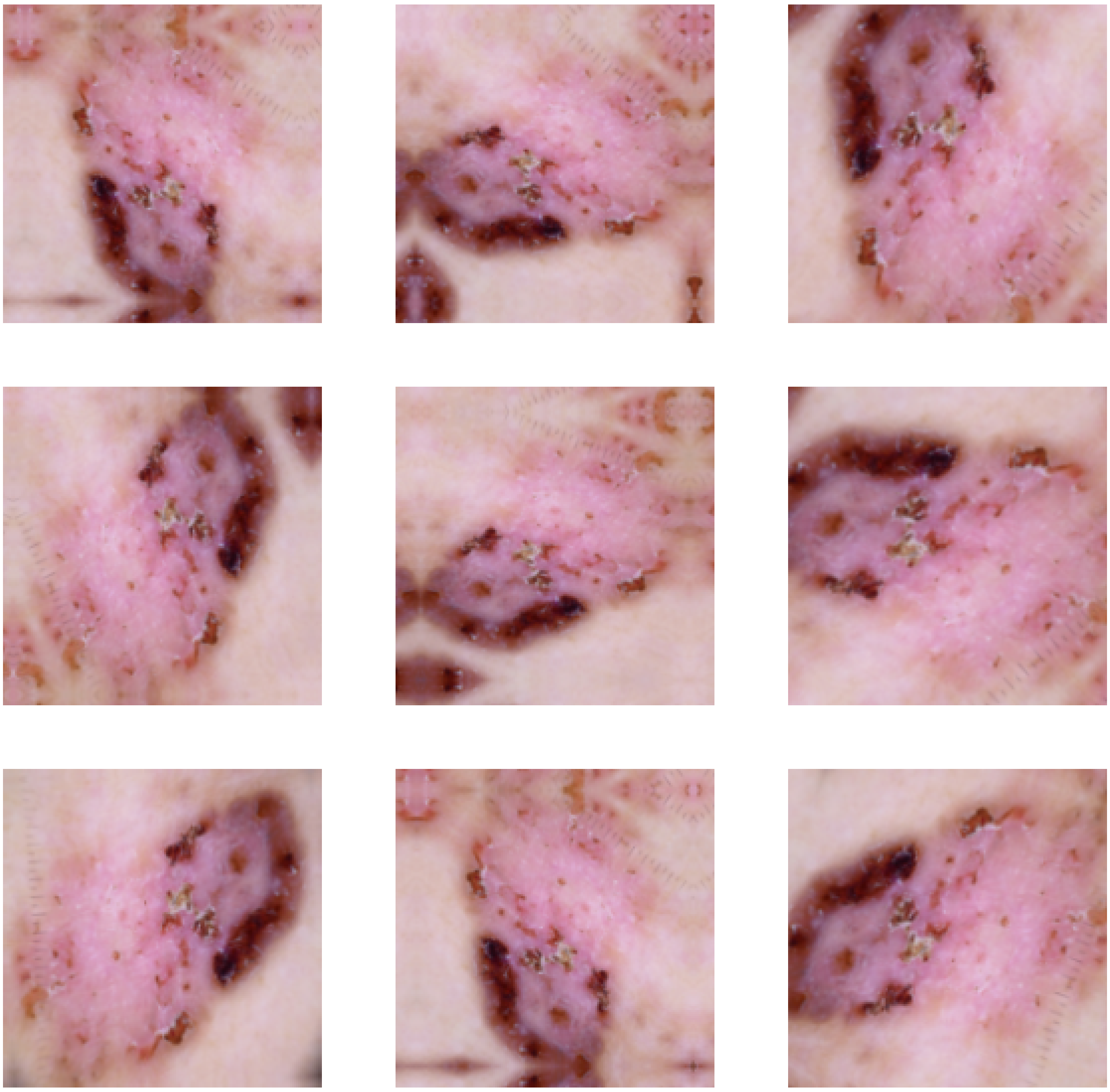
```
#data augumentation strategy.

data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical",seed=123),
  layers.RandomRotation(0.2,seed=123),
  layers.RandomZoom(0.2,seed=123)
])
```

In [20]:

```
plt.figure(figsize=(10, 10))

for images,labels in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    augmented_image = augmented_images[0].numpy().astype('uint8')
    plt.imshow(augmented_image)
    plt.axis("off")
```



Create the model, compile and train the model

In [24]:

```
model2 = Sequential()
model2.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(18
model2.add(data_augmentation)

model2.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shap
model2.add(MaxPool2D(pool_size=(2, 2)))

model2.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu'))
model2.add(MaxPool2D(pool_size=(2, 2)))

model2.add(Conv2D(128, kernel_size=(3, 3), padding='same', activation='relu'))
model2.add(MaxPool2D(pool_size=(2, 2)))

model2.add(Flatten())

model2.add(Dense(512, activation='relu'))
model2.add(Dropout(0.5))
model2.add(Dense(9, activation='softmax'))
```

Compiling the model

In [25]:

```
model2.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [26]:

```
model2.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 module_wrapper_20 (ModuleWr  (None, 180, 180, 32)     896       
 apper)                                                          
                                                                 
 module_wrapper_21 (ModuleWr  (None, 90, 90, 32)       0         
 apper)                                                          
                                                                 
 module_wrapper_22 (ModuleWr  (None, 90, 90, 64)       18496     
 apper)                                                          
                                                                 
 module_wrapper_23 (ModuleWr  (None, 45, 45, 64)       0         
 apper)                                                          
                                                                 
 module_wrapper_24 (ModuleWr  (None, 45, 45, 128)      73856     
 apper)                                                          
                                                                 
 module_wrapper_25 (ModuleWr  (None, 22, 22, 128)      0         
 apper)                                                          
                                                                 
 module_wrapper_26 (ModuleWr  (None, 61952)            0         
 apper)                                                          
                                                                 
 module_wrapper_27 (ModuleWr  (None, 512)              31719936  
 apper)                                                          
                                                                 
 module_wrapper_28 (ModuleWr  (None, 512)              0         
 apper)                                                          
                                                                 
 module_wrapper_29 (ModuleWr  (None, 9)                4617      
 apper)                                                          
                                                                 
=================================================================
Total params: 31,817,801
Trainable params: 31,817,801
Non-trainable params: 0
_________________________________________________________________
```

Training the model

In [27]:

```
epochs = 20
history = model2.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 20s 340ms/step - loss: 2.2378 - acc
uracy: 0.2260 - val_loss: 1.8831 - val_accuracy: 0.3423
Epoch 2/20
56/56 [==============================] - 19s 337ms/step - loss: 1.8279 - acc
uracy: 0.3287 - val_loss: 1.9789 - val_accuracy: 0.2796
Epoch 3/20
56/56 [==============================] - 19s 340ms/step - loss: 1.6545 - acc
uracy: 0.3990 - val_loss: 1.4980 - val_accuracy: 0.4541
Epoch 4/20
56/56 [==============================] - 19s 337ms/step - loss: 1.4750 - acc
uracy: 0.4682 - val_loss: 1.4584 - val_accuracy: 0.5056
Epoch 5/20
56/56 [==============================] - 19s 335ms/step - loss: 1.4561 - acc
uracy: 0.4939 - val_loss: 1.4754 - val_accuracy: 0.5190
Epoch 6/20
56/56 [==============================] - 19s 338ms/step - loss: 1.4255 - acc
uracy: 0.5017 - val_loss: 1.4349 - val_accuracy: 0.5190
Epoch 7/20
56/56 [==============================] - 19s 339ms/step - loss: 1.4163 - acc
uracy: 0.4888 - val_loss: 1.3736 - val_accuracy: 0.5145
Epoch 8/20
56/56 [==============================] - 19s 337ms/step - loss: 1.3660 - acc
uracy: 0.5206 - val_loss: 1.3844 - val_accuracy: 0.5257
Epoch 9/20
56/56 [==============================] - 19s 338ms/step - loss: 1.3468 - acc
uracy: 0.5134 - val_loss: 1.4100 - val_accuracy: 0.5190
Epoch 10/20
56/56 [==============================] - 19s 335ms/step - loss: 1.3113 - acc
uracy: 0.5329 - val_loss: 1.3691 - val_accuracy: 0.5235
Epoch 11/20
56/56 [==============================] - 19s 339ms/step - loss: 1.2966 - acc
uracy: 0.5491 - val_loss: 1.3827 - val_accuracy: 0.5280
Epoch 12/20
56/56 [==============================] - 19s 334ms/step - loss: 1.2924 - acc
uracy: 0.5301 - val_loss: 1.3570 - val_accuracy: 0.5034
Epoch 13/20
56/56 [==============================] - 19s 336ms/step - loss: 1.3088 - acc
uracy: 0.5290 - val_loss: 1.3094 - val_accuracy: 0.5414
Epoch 14/20
56/56 [==============================] - 19s 340ms/step - loss: 1.2615 - acc
uracy: 0.5413 - val_loss: 1.3973 - val_accuracy: 0.5369
Epoch 15/20
56/56 [==============================] - 19s 335ms/step - loss: 1.2704 - acc
uracy: 0.5352 - val_loss: 1.3768 - val_accuracy: 0.5280
Epoch 16/20
56/56 [==============================] - 19s 336ms/step - loss: 1.2257 - acc
uracy: 0.5575 - val_loss: 1.3611 - val_accuracy: 0.5548
Epoch 17/20
56/56 [==============================] - 19s 339ms/step - loss: 1.2304 - acc
uracy: 0.5569 - val_loss: 1.5553 - val_accuracy: 0.5145
Epoch 18/20
```

```
56/56 [==============================] - 19s 337ms/step - loss: 1.2495 - acc
uracy: 0.5552 - val_loss: 1.4216 - val_accuracy: 0.5101
Epoch 19/20
56/56 [==============================] - 19s 340ms/step - loss: 1.2429 - acc
uracy: 0.5552 - val_loss: 1.3139 - val_accuracy: 0.5369
Epoch 20/20
56/56 [==============================] - 19s 340ms/step - loss: 1.2442 - acc
uracy: 0.5592 - val_loss: 1.3109 - val_accuracy: 0.5302
```

Visualizing the results

In [28]:

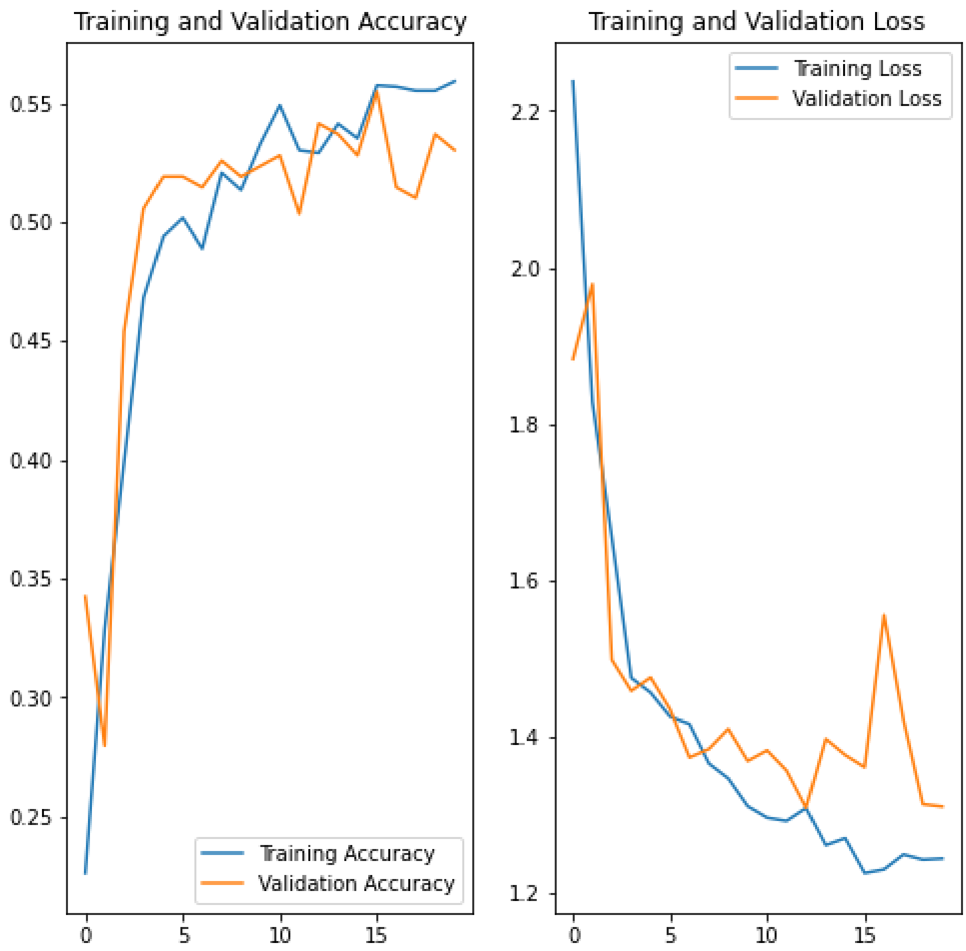
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Observation Model 2 - The gap in accuracy of training and validation has now reduced which means the issue of over fitting is resolved. However, now both training and validation accuracy is low around 55% which means the model is underfitting.

Model 3 : Class Imbalance Rectification using Augmentor

Context: Many times real life datasets can have class imbalance, one class can have proportionately higher number of samples compared to the others. Class imbalance can have a detrimental effect on the final model quality. Hence as a sanity check it becomes important to check what is the distribution of classes in the data.

In [29]:

```
data = dict()
for i in class_names:
  data[i] = 0

for images,labels in train_ds:
  l=labels.numpy()
  s = sum(l)
  for i in range(9):
    data[class_names[i]] = data[class_names[i]]+s[i]

data
```

Out[29]:

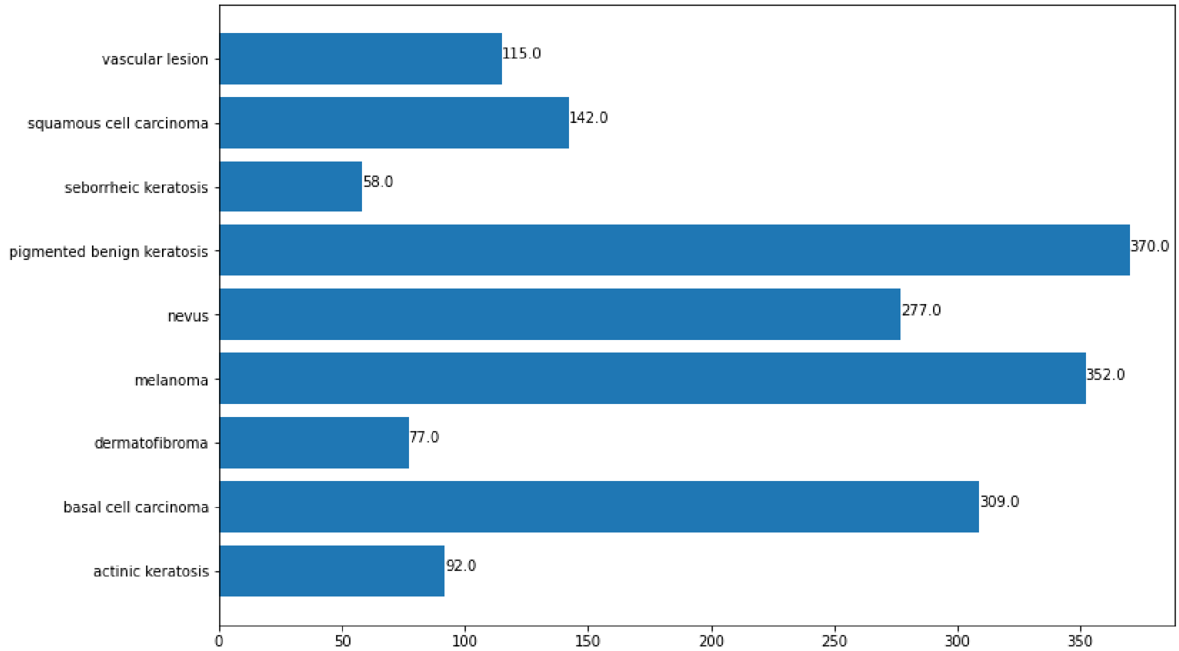
```
{'actinic keratosis': 92.0,
 'basal cell carcinoma': 309.0,
 'dermatofibroma': 77.0,
 'melanoma': 352.0,
 'nevus': 277.0,
 'pigmented benign keratosis': 370.0,
 'seborrheic keratosis': 58.0,
 'squamous cell carcinoma': 142.0,
 'vascular lesion': 115.0}
```

In [30]:

```
plt.figure(figsize=(12, 8))
plt.barh(range(len(data)), list(data.values()))
plt.yticks(range(len(data)), list(data.keys()))

for index, value in enumerate(data.values()):
    plt.text(value, index,
             str(value))

plt.show()
```



Observation: There is a class imbalance in the train data set.

- seborrheic keratosis, actinic keratosis, dermatofibroma classes have low number of data samples.

- Whereas, melanoma, basal cell carcinoma, pigmented benign keratosis have higher samples.

Rectify the class imbalance

Context: You can use a python package known as `Augmentor` ([https://augmentor.readthedocs.io/en/master/ (https://augmentor.readthedocs.io/en/master/)](https://augmentor.readthedocs.io/en/master/)) to add more samples across all classes so that none of the classes have very few samples.

In [31]:

```
!pip install Augmentor
```

```
Collecting Augmentor
  Downloading Augmentor-0.2.9-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dis
t-packages (from Augmentor) (1.21.6)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.7/dis
t-packages (from Augmentor) (7.1.2)
Requirement already satisfied: future>=0.16.0 in /usr/local/lib/python3.7/di
st-packages (from Augmentor) (0.16.0)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.7/dist-
packages (from Augmentor) (4.64.0)
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.9
```

To use `Augmentor` , the following general procedure is followed:

1. Instantiate a `Pipeline` object pointing to a directory containing your initial image data set.
2. Define a number of operations to perform on this data set using your `Pipeline` object.
3. Execute these operations by calling the `Pipeline’s sample()` method.

In [32]:

```
root_path = '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data'
data_dir_train = pathlib.Path(root_path + '/Train')
data_dir_test = pathlib.Path(root_path + '/Test')

path_to_training_dataset = pathlib.Path(str(data_dir_train) + '/')
import Augmentor
for i in class_names:
    path_to_training_dataset = pathlib.Path(str(data_dir_train) + '/' + i)
    p = Augmentor.Pipeline(path_to_training_dataset)
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(500) ## We are adding 500 samples per class to make sure that none of the
```

```
Initialised with 114 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/actinic keratosis/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at
0x7F2C38885910>: 100%|██████████| 500/500 [00:02<00:00, 189.42 Samples/s]

Initialised with 376 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/basal cell carcinoma/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at
0x7F2C38888650>: 100%|██████████| 500/500 [00:02<00:00, 178.37 Samples/s]

Initialised with 95 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/dermatofibroma/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7F2C4A310E50>:
100%|██████████| 500/500 [00:02<00:00, 182.15 Samples/s]

Initialised with 438 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/melanoma/output.

Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x7F2C49F5A290>:
100%|██████████| 500/500 [00:08<00:00, 61.97 Samples/s]

Initialised with 357 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/nevus/output.

Processing <PIL.Image.Image image mode=RGB size=2048x1536 at 0x7F2C4AB88850
>: 100%|██████████| 500/500 [00:07<00:00, 68.62 Samples/s]

Initialised with 462 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/pigmented benign keratosis/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7F2C388F5F90>:
100%|██████████| 500/500 [00:02<00:00, 178.48 Samples/s]

Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/seborrheic keratosis/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1024x768 a
t 0x7F2C38538810>: 100%|██████████| 500/500 [00:03<00:00, 130.89 Samples/s]

Initialised with 181 image(s) found.
```

```
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melano
ma/Data/Train/squamous cell carcinoma/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at
0x7F2C4AAC2090>: 100%|██████████| 500/500 [00:02<00:00, 173.24 Samples/s]

Initialised with 139 image(s) found.
Output directory set to /content/gdrive/MyDrive/Colab Notebooks/CNN_Melanom
a/Data/Train/vascular lesion/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7F2C38948810>:
100%|██████████| 500/500 [00:03<00:00, 162.38 Samples/s]
```

Augmentor has stored the augmented images in the output sub-directory of each of the sub-directories of skin cancer types.. Lets take a look at total count of augmented images.

In [33]:

```
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg')))
print(image_count_train)
```

```
4500
```

Lets see the distribution of augmented data after adding new images to the original training data.

In [34]:

```
path_list_new = [x for x in glob(os.path.join(data_dir_train, '*','output', '*.jpg'))]
path_list_new
```

Out[34]:

```
['/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0030133.jpg_35ade46f-c2f8
-4bea-a551-b0b4d5056624.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0032404.jpg_0c3dd298-27f2
-4898-b93f-6d04b71915d4.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0025953.jpg_d2de699d-5ee4
-46bb-8ad5-a1134692056c.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0028393.jpg_0ceb2458-baf1
-421a-8096-5a9c1bda7b97.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0026040.jpg_be65f2ef-a5f7
-4b9b-821e-e23bd4e503dd.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/output/actinic keratosis_original_ISIC_0027580.jpg_915b603f-007d
-46a2-b00e-dc9099d9b7fe.jpg',
```

In [35]:

```
path_list_old = [x for x in glob(os.path.join(data_dir_train, '*','*.jpg'))]
path_list_old
```

Out[35]:

```
['/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026857.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0025957.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026194.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026457.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026040.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026171.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026575.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026525.jpg',
 '/content/gdrive/MyDrive/Colab Notebooks/CNN_Melanoma/Data/Train/actinic
keratosis/ISIC_0026848.jpg',
```

In [36]:

```
lesion_list_new = [os.path.basename(os.path.dirname(os.path.dirname(y))) for y in glob(
lesion_list_new
```

Out[36]:

```
['actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
```

In [37]:

```
lesion_list_old = [os.path.basename(os.path.dirname(y)) for y in glob(os.path.join(data
lesion_list_old
```

Out[37]:

```
['actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
```

In [38]:

```
dataframe_dict_new = dict(zip(path_list_new, lesion_list_new))
```

In [39]:

```
dataframe_dict_old = dict(zip(path_list_old, lesion_list_old))
```

In [40]:

```
df2 = pd.DataFrame(list(dataframe_dict_new.items()),columns = ['Path','Label'])
original_df = pd.DataFrame(list(dataframe_dict_old.items()),columns = ['Path','Label'])
new_df = original_df.append(df2)
```

In [41]:

```
new_df['Label'].value_counts()
```

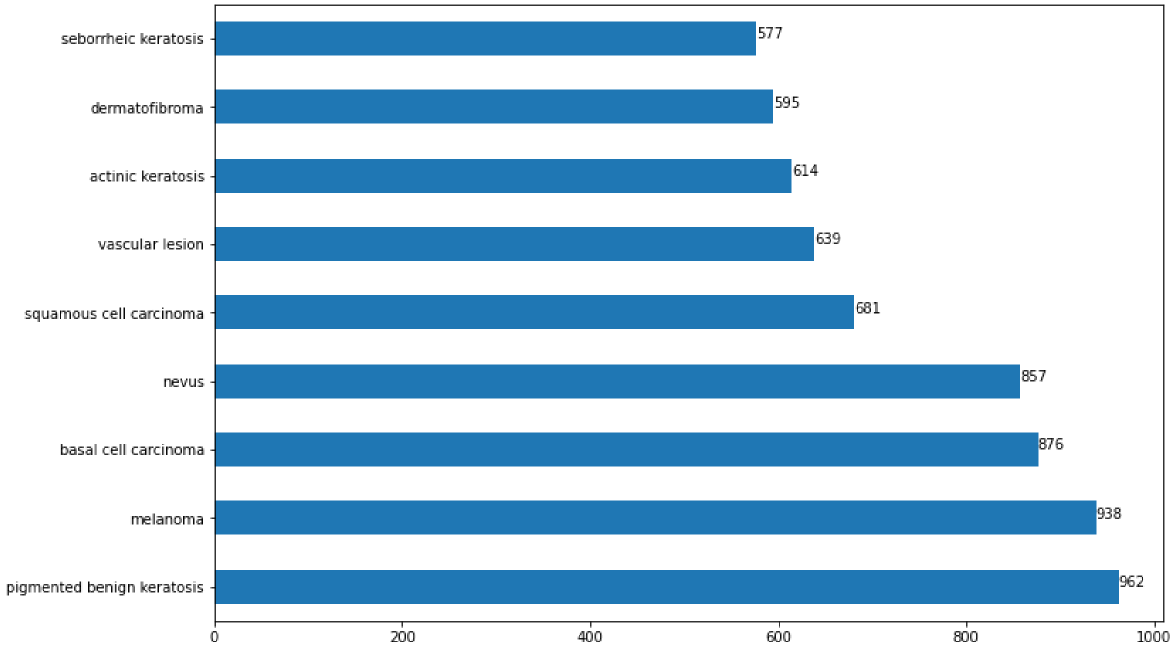
Out[41]:

```
pigmented benign keratosis    962
melanoma                      938
basal cell carcinoma          876
nevus                         857
squamous cell carcinoma       681
vascular lesion               639
actinic keratosis             614
dermatofibroma                595
seborrheic keratosis          577
Name: Label, dtype: int64
```

In [42]:

```
plt.figure(figsize=(12, 8))
new_df['Label'].value_counts().plot(kind = 'barh')
for index, value in enumerate(new_df['Label'].value_counts()):
    plt.text(value, index,
             str(value))

plt.show()
```



Observation : So, now we have added 500 images to all the classes to maintain some class balance. We can add more images as we want to improve training process.

Train the model on the data created using Augmentor

In [43]:

```
batch_size = 32
img_height = 180
img_width = 180
```

Create a training dataset

In [44]:

```
#data_dir_train="path to directory with training data + data created using augmentor"
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.2,
  labels='inferred',
  subset="training",
  label_mode='categorical',
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 6739 files belonging to 9 classes.
Using 5392 files for training.
```

Todo: Create a validation dataset

In [45]:

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  validation_split=0.2,
  labels='inferred',
  subset="validation",
  label_mode='categorical',
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 6739 files belonging to 9 classes.
Using 1347 files for validation.
```

Todo: Create your model (make sure to include normalization)

The model is built with batch normalisation

In [46]:

```
model = Sequential()

model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(18(

model.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu', input_shape
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(9, activation='softmax'))
```

In [47]:

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:

```
epochs = 30
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/30
169/169 [==============================] - 9s 44ms/step - loss: 2.2656 - acc
uracy: 0.3221 - val_loss: 2.9874 - val_accuracy: 0.1151
Epoch 2/30
169/169 [==============================] - 7s 43ms/step - loss: 1.7377 - acc
uracy: 0.4089 - val_loss: 2.9838 - val_accuracy: 0.2108
Epoch 3/30
169/169 [==============================] - 7s 43ms/step - loss: 1.5021 - acc
uracy: 0.4726 - val_loss: 2.4183 - val_accuracy: 0.2687
Epoch 4/30
169/169 [==============================] - 7s 44ms/step - loss: 1.3476 - acc
uracy: 0.5080 - val_loss: 1.3319 - val_accuracy: 0.5152
Epoch 5/30
169/169 [==============================] - 7s 44ms/step - loss: 1.2010 - acc
uracy: 0.5584 - val_loss: 1.2250 - val_accuracy: 0.5241
Epoch 6/30
169/169 [==============================] - 7s 43ms/step - loss: 1.0835 - acc
uracy: 0.6129 - val_loss: 1.2442 - val_accuracy: 0.5375
Epoch 7/30
169/169 [==============================] - 7s 43ms/step - loss: 0.9914 - acc
uracy: 0.6309 - val_loss: 1.7882 - val_accuracy: 0.4254
Epoch 8/30
169/169 [==============================] - 7s 43ms/step - loss: 0.8492 - acc
uracy: 0.6851 - val_loss: 1.0294 - val_accuracy: 0.6310
Epoch 9/30
169/169 [==============================] - 7s 43ms/step - loss: 0.7789 - acc
uracy: 0.7140 - val_loss: 1.8155 - val_accuracy: 0.4670
Epoch 10/30
169/169 [==============================] - 7s 43ms/step - loss: 0.6729 - acc
uracy: 0.7567 - val_loss: 1.8271 - val_accuracy: 0.4662
Epoch 11/30
169/169 [==============================] - 7s 43ms/step - loss: 0.5874 - acc
uracy: 0.7854 - val_loss: 1.9502 - val_accuracy: 0.4714
Epoch 12/30
169/169 [==============================] - 7s 43ms/step - loss: 0.5576 - acc
uracy: 0.7858 - val_loss: 2.7794 - val_accuracy: 0.3474
Epoch 13/30
169/169 [==============================] - 7s 43ms/step - loss: 0.4639 - acc
uracy: 0.8281 - val_loss: 1.2510 - val_accuracy: 0.6244
Epoch 14/30
169/169 [==============================] - 7s 43ms/step - loss: 0.4053 - acc
uracy: 0.8481 - val_loss: 1.7808 - val_accuracy: 0.5271
Epoch 15/30
169/169 [==============================] - 7s 43ms/step - loss: 0.4160 - acc
uracy: 0.8451 - val_loss: 1.2760 - val_accuracy: 0.5820
Epoch 16/30
169/169 [==============================] - 7s 43ms/step - loss: 0.3359 - acc
uracy: 0.8704 - val_loss: 1.4344 - val_accuracy: 0.5590
Epoch 17/30
169/169 [==============================] - 7s 43ms/step - loss: 0.3389 - acc
uracy: 0.8687 - val_loss: 1.1254 - val_accuracy: 0.6585
Epoch 18/30
```

```
169/169 [==============================] - 7s 43ms/step - loss: 0.2474 - acc
uracy: 0.9058 - val_loss: 0.6937 - val_accuracy: 0.7803
Epoch 19/30
169/169 [==============================] - 7s 43ms/step - loss: 0.2187 - acc
uracy: 0.9190 - val_loss: 0.8971 - val_accuracy: 0.7246
Epoch 20/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1957 - acc
uracy: 0.9256 - val_loss: 1.0917 - val_accuracy: 0.6763
Epoch 21/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1917 - acc
uracy: 0.9318 - val_loss: 1.0376 - val_accuracy: 0.6956
Epoch 22/30
169/169 [==============================] - 7s 43ms/step - loss: 0.2058 - acc
uracy: 0.9177 - val_loss: 1.3957 - val_accuracy: 0.6288
Epoch 23/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1966 - acc
uracy: 0.9243 - val_loss: 1.6049 - val_accuracy: 0.5872
Epoch 24/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1698 - acc
uracy: 0.9327 - val_loss: 0.9941 - val_accuracy: 0.7305
Epoch 25/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1626 - acc
uracy: 0.9403 - val_loss: 0.8175 - val_accuracy: 0.7661
Epoch 26/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1575 - acc
uracy: 0.9392 - val_loss: 0.7186 - val_accuracy: 0.7884
Epoch 27/30
169/169 [==============================] - 7s 44ms/step - loss: 0.1422 - acc
uracy: 0.9483 - val_loss: 1.0482 - val_accuracy: 0.7320
Epoch 28/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1431 - acc
uracy: 0.9484 - val_loss: 0.9466 - val_accuracy: 0.7120
Epoch 29/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1433 - acc
uracy: 0.9466 - val_loss: 1.4261 - val_accuracy: 0.6258
Epoch 30/30
169/169 [==============================] - 7s 43ms/step - loss: 0.1489 - acc
uracy: 0.9434 - val_loss: 1.1115 - val_accuracy: 0.7149
```

In [ ]:

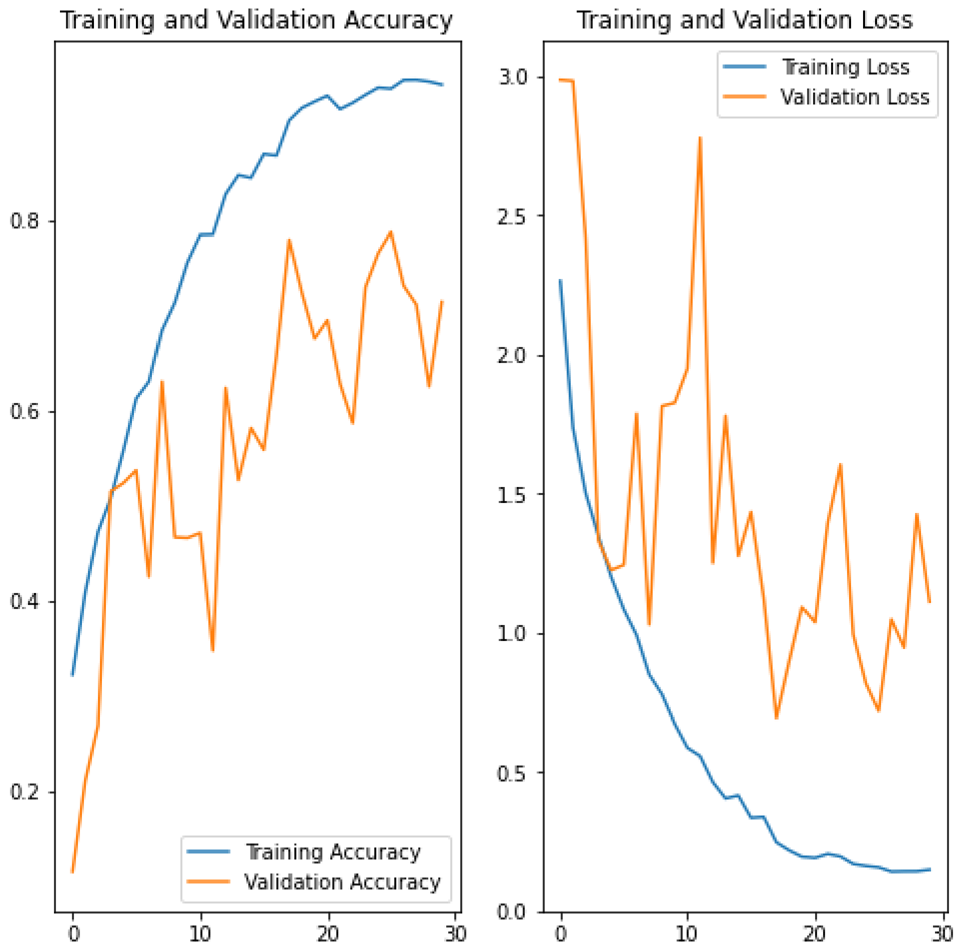
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



We observed the model is still overfitting to some extent and some erratic behaviour. The above model is rebuilt removing the batch normalisation steps.

In [49]:

```
model3 = Sequential()

model3.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(18

model3.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu', input_shap
model3.add(MaxPool2D(pool_size=(2, 2)))
model3.add(Dropout(0.25))

model3.add(Conv2D(64, kernel_size=(3, 3), padding='same', activation='relu'))
model3.add(MaxPool2D(pool_size=(2, 2)))
model3.add(Dropout(0.25))

model3.add(Conv2D(128, kernel_size=(3, 3), padding='same', activation='relu'))
model3.add(MaxPool2D(pool_size=(2, 2)))
model3.add(Dropout(0.25))

model3.add(Flatten())

model3.add(Dense(256, activation='relu'))
model3.add(Dropout(0.5))
model3.add(Dense(512, activation='relu'))
model3.add(Dropout(0.5))
model3.add(Dense(9, activation='softmax'))
```

Todo: Compile your model (Choose optimizer and loss function appropriately)

In [50]:

```
model3.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [51]:

```
model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_4 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 module_wrapper_45 (ModuleWr  (None, 180, 180, 64)     1792      
 apper)                                                          
                                                                 
 module_wrapper_46 (ModuleWr  (None, 90, 90, 64)       0         
 apper)                                                          
                                                                 
 module_wrapper_47 (ModuleWr  (None, 90, 90, 64)       0         
 apper)                                                          
                                                                 
 module_wrapper_48 (ModuleWr  (None, 90, 90, 64)       36928     
 apper)                                                          
                                                                 
 module_wrapper_49 (ModuleWr  (None, 45, 45, 64)       0         
 apper)                                                          
                                                                 
 module_wrapper_50 (ModuleWr  (None, 45, 45, 64)       0         
 apper)                                                          
                                                                 
 module_wrapper_51 (ModuleWr  (None, 45, 45, 128)      73856     
 apper)                                                          
                                                                 
 module_wrapper_52 (ModuleWr  (None, 22, 22, 128)      0         
 apper)                                                          
                                                                 
 module_wrapper_53 (ModuleWr  (None, 22, 22, 128)      0         
 apper)                                                          
                                                                 
 module_wrapper_54 (ModuleWr  (None, 61952)            0         
 apper)                                                          
                                                                 
 module_wrapper_55 (ModuleWr  (None, 256)              15859968  
 apper)                                                          
                                                                 
 module_wrapper_56 (ModuleWr  (None, 256)              0         
 apper)                                                          
                                                                 
 module_wrapper_57 (ModuleWr  (None, 512)              131584    
 apper)                                                          
                                                                 
 module_wrapper_58 (ModuleWr  (None, 512)              0         
 apper)                                                          
                                                                 
 module_wrapper_59 (ModuleWr  (None, 9)                4617      
 apper)                                                          
                                                                 
=================================================================
Total params: 16,108,745
Trainable params: 16,108,745
Non-trainable params: 0
_________________________________________________________________
```

Todo: Train your model

In [52]:

```
epochs = 30
history = model3.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/30
169/169 [==============================] - 84s 493ms/step - loss: 2.2566 - a
ccuracy: 0.1317 - val_loss: 2.1865 - val_accuracy: 0.1403
Epoch 2/30
169/169 [==============================] - 82s 485ms/step - loss: 2.1781 - a
ccuracy: 0.1398 - val_loss: 2.1873 - val_accuracy: 0.1403
Epoch 3/30
169/169 [==============================] - 83s 488ms/step - loss: 2.1755 - a
ccuracy: 0.1397 - val_loss: 2.1881 - val_accuracy: 0.1403
Epoch 4/30
169/169 [==============================] - 81s 478ms/step - loss: 2.1777 - a
ccuracy: 0.1411 - val_loss: 2.1840 - val_accuracy: 0.1403
Epoch 5/30
169/169 [==============================] - 80s 473ms/step - loss: 2.1750 - a
ccuracy: 0.1439 - val_loss: 2.0840 - val_accuracy: 0.2160
Epoch 6/30
169/169 [==============================] - 80s 471ms/step - loss: 2.0684 - a
ccuracy: 0.1992 - val_loss: 1.8468 - val_accuracy: 0.3177
Epoch 7/30
169/169 [==============================] - 80s 471ms/step - loss: 1.9089 - a
ccuracy: 0.2619 - val_loss: 1.6510 - val_accuracy: 0.3682
Epoch 8/30
169/169 [==============================] - 80s 471ms/step - loss: 1.6506 - a
ccuracy: 0.3741 - val_loss: 1.4708 - val_accuracy: 0.4514
Epoch 9/30
169/169 [==============================] - 80s 474ms/step - loss: 1.4872 - a
ccuracy: 0.4149 - val_loss: 1.3787 - val_accuracy: 0.4774
Epoch 10/30
169/169 [==============================] - 80s 475ms/step - loss: 1.4430 - a
ccuracy: 0.4483 - val_loss: 1.2986 - val_accuracy: 0.5226
Epoch 11/30
169/169 [==============================] - 79s 469ms/step - loss: 1.3640 - a
ccuracy: 0.4740 - val_loss: 1.4038 - val_accuracy: 0.4818
Epoch 12/30
169/169 [==============================] - 79s 469ms/step - loss: 1.3130 - a
ccuracy: 0.4909 - val_loss: 1.1790 - val_accuracy: 0.5516
Epoch 13/30
169/169 [==============================] - 80s 470ms/step - loss: 1.1975 - a
ccuracy: 0.5458 - val_loss: 1.1035 - val_accuracy: 0.5902
Epoch 14/30
169/169 [==============================] - 80s 469ms/step - loss: 1.1541 - a
ccuracy: 0.5616 - val_loss: 1.1554 - val_accuracy: 0.5739
Epoch 15/30
169/169 [==============================] - 80s 473ms/step - loss: 1.0723 - a
ccuracy: 0.5994 - val_loss: 1.0655 - val_accuracy: 0.6013
Epoch 16/30
169/169 [==============================] - 81s 478ms/step - loss: 0.9706 - a
ccuracy: 0.6319 - val_loss: 0.8694 - val_accuracy: 0.7201
Epoch 17/30
169/169 [==============================] - 80s 474ms/step - loss: 0.9393 - a
ccuracy: 0.6448 - val_loss: 0.8834 - val_accuracy: 0.6823
Epoch 18/30
```

```
169/169 [==============================] - 80s 471ms/step - loss: 0.8633 - a
ccuracy: 0.6753 - val_loss: 0.8718 - val_accuracy: 0.6860
Epoch 19/30
169/169 [==============================] - 80s 473ms/step - loss: 0.7898 - a
ccuracy: 0.7027 - val_loss: 0.7496 - val_accuracy: 0.7209
Epoch 20/30
169/169 [==============================] - 80s 469ms/step - loss: 0.7598 - a
ccuracy: 0.7214 - val_loss: 0.7091 - val_accuracy: 0.7305
Epoch 21/30
169/169 [==============================] - 81s 476ms/step - loss: 0.7551 - a
ccuracy: 0.7261 - val_loss: 0.7316 - val_accuracy: 0.7394
Epoch 22/30
169/169 [==============================] - 80s 471ms/step - loss: 0.6468 - a
ccuracy: 0.7552 - val_loss: 0.6442 - val_accuracy: 0.7617
Epoch 23/30
169/169 [==============================] - 80s 473ms/step - loss: 0.6002 - a
ccuracy: 0.7752 - val_loss: 0.6625 - val_accuracy: 0.7565
Epoch 24/30
169/169 [==============================] - 80s 471ms/step - loss: 0.5505 - a
ccuracy: 0.7969 - val_loss: 0.6308 - val_accuracy: 0.7840
Epoch 25/30
169/169 [==============================] - 80s 473ms/step - loss: 0.5837 - a
ccuracy: 0.7787 - val_loss: 0.6241 - val_accuracy: 0.7743
Epoch 26/30
169/169 [==============================] - 80s 471ms/step - loss: 0.5174 - a
ccuracy: 0.8047 - val_loss: 0.6200 - val_accuracy: 0.7751
Epoch 27/30
169/169 [==============================] - 81s 480ms/step - loss: 0.5032 - a
ccuracy: 0.8194 - val_loss: 0.6341 - val_accuracy: 0.7869
Epoch 28/30
169/169 [==============================] - 84s 494ms/step - loss: 0.4801 - a
ccuracy: 0.8192 - val_loss: 0.5688 - val_accuracy: 0.8040
Epoch 29/30
169/169 [==============================] - 84s 494ms/step - loss: 0.4558 - a
ccuracy: 0.8240 - val_loss: 0.5715 - val_accuracy: 0.8092
Epoch 30/30
169/169 [==============================] - 84s 496ms/step - loss: 0.4486 - a
ccuracy: 0.8316 - val_loss: 0.5215 - val_accuracy: 0.8166
```

Todo: Visualize the model results

In [53]:

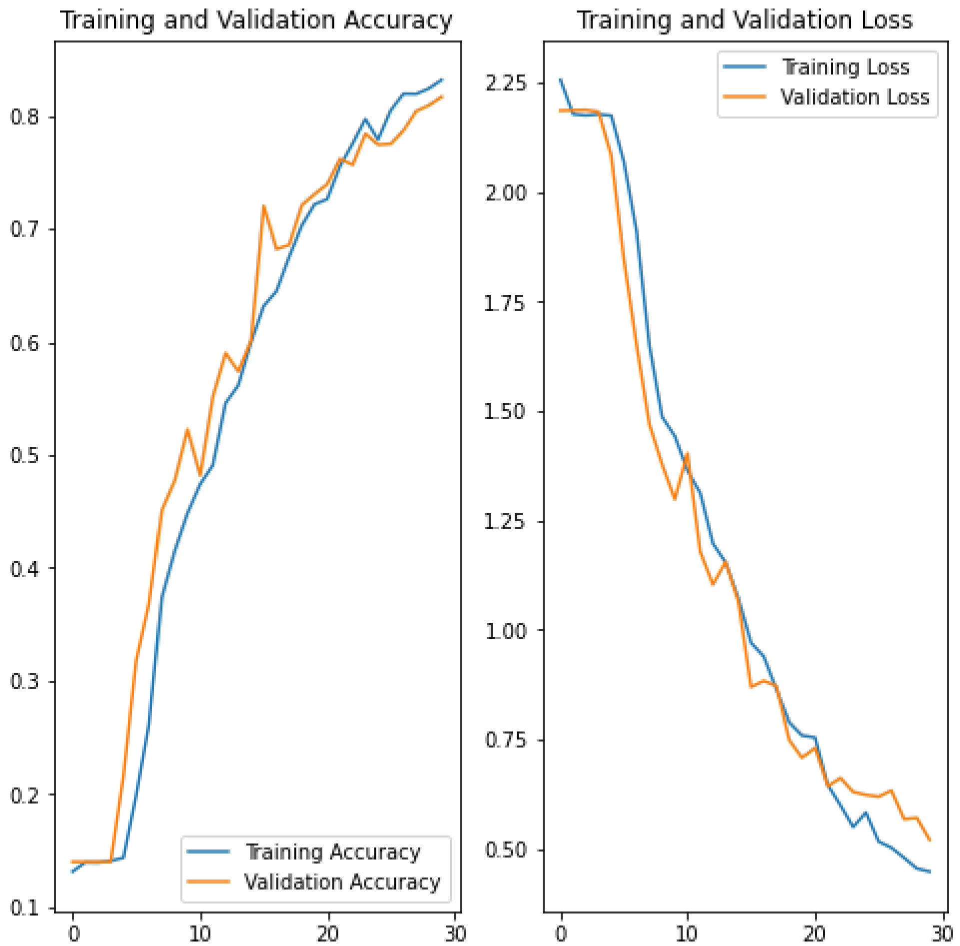
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Did you get rid of underfitting/overfitting? Did class rebalance help?

Observation Model 3 - With the final model, the issue of overfitting in model 1 and underfitting in model 2 has been resolved. The classes are now more balaced. The accuracy is around 83% and there is not major gap between training and validation accuracy.