Detect Melanoma

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.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

1. Import Libraries

```
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from glob import glob
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization, Conv2D, MaxPooling2D
```

```
train_path="/content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train"
test_path="/content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Test"

data_dir_train = pathlib.Path(train_path)
data_dir_test = pathlib.Path(test_path)
```

```
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
```

2. Data Preparation

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

```
img_width = 180

# Train Data Set Creation
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train, labels='inferred', label_mode='categorical',
    class_names=None, color_mode='rgb', batch_size=32, image_size=(180,
    180), shuffle=True, seed=123, validation_split=0.2, subset='training',
    interpolation='bilinear', follow_links=False, smart_resize=False
)
```

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

```
# Validation Data Set Creation
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train, labels='inferred', label_mode='categorical',
    class_names=None, color_mode='rgb', batch_size=32, image_size=(180,
    180), shuffle=True, seed=123, validation_split=0.2, subset='validation',
    interpolation='bilinear', follow_links=False, smart_resize=False
)
```

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

```
class_names = train_ds.class_names
print(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', 'nevus', 'pigmented benign keratosis', 'seborrheic keratosis', 'squamous cell carcinoma', 'vascular les
```

3. Visualize the data

```
import matplotlib.pyplot as plt
num=0
for dirpath, dirnames, filenames in os.walk(str(train_path)):
    for filename in [f for f in filenames if f.endswith(".jpg")][:1]:
        img = PIL.Image.open(str(dirpath)+"/"+str(filename))
        plt.subplot(3,3,num+1)
        plt.title(str(dirpath).split('/')[-1])
        plt.axis('off')
        plt.imshow(img)
        num=num+1
```


```
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)
```

4. Model Creation

Model 0

```
# Creating the model
model=Sequential([
    tf.keras.layers.experimental.preprocessing.Rescaling(scale=1./255., offset=0.0,),

    Conv2D(32,(3,3),input_shape=(img_height,img_width,3),activation='relu',padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.1),

    Conv2D(64,(3,3),activation='relu',padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.1),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(9, activation='softmax')
])


#Compiling the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])


#Training the model
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 408s 1s/step - loss: 3.1948 - accuracy: 0.1775 - val_loss: 2.1011 - val_accuracy: 0.1902
Epoch 2/20
56/56 [==============================] - 2s 40ms/step - loss: 1.9948 - accuracy: 0.2372 - val_loss: 1.9514 - val_accuracy: 0.2998
Epoch 3/20
56/56 [==============================] - 2s 39ms/step - loss: 1.8306 - accuracy: 0.3359 - val_loss: 1.6695 - val_accuracy: 0.4161
Epoch 4/20
```

```
56/56 [==============================] - 2s 39ms/step - loss: 1.6174 - accuracy: 0.4342 - val_loss: 1.5905 - val_accuracy: 0.4295
Epoch 5/20
56/56 [==============================] - 2s 39ms/step - loss: 1.4830 - accuracy: 0.4855 - val_loss: 1.5229 - val_accuracy: 0.4810
Epoch 6/20
56/56 [==============================] - 2s 39ms/step - loss: 1.3810 - accuracy: 0.5145 - val_loss: 1.4469 - val_accuracy: 0.5123
Epoch 7/20
56/56 [==============================] - 2s 39ms/step - loss: 1.3159 - accuracy: 0.5452 - val_loss: 1.4801 - val_accuracy: 0.4966
Epoch 8/20
56/56 [==============================] - 2s 39ms/step - loss: 1.2254 - accuracy: 0.5625 - val_loss: 1.4279 - val_accuracy: 0.5213
Epoch 9/20
56/56 [==============================] - 2s 39ms/step - loss: 1.1453 - accuracy: 0.6021 - val_loss: 1.3924 - val_accuracy: 0.5257
Epoch 10/20
56/56 [==============================] - 2s 39ms/step - loss: 1.0855 - accuracy: 0.6183 - val_loss: 1.4127 - val_accuracy: 0.5324
Epoch 11/20
56/56 [==============================] - 2s 39ms/step - loss: 0.9949 - accuracy: 0.6412 - val_loss: 1.4356 - val_accuracy: 0.5302
Epoch 12/20
56/56 [==============================] - 2s 39ms/step - loss: 0.9243 - accuracy: 0.6713 - val_loss: 1.5570 - val_accuracy: 0.5347
Epoch 13/20
56/56 [==============================] - 2s 39ms/step - loss: 0.8647 - accuracy: 0.6853 - val_loss: 1.5601 - val_accuracy: 0.5436
Epoch 14/20
56/56 [==============================] - 2s 39ms/step - loss: 0.7598 - accuracy: 0.7321 - val_loss: 1.6101 - val_accuracy: 0.5347
Epoch 15/20
56/56 [==============================] - 2s 39ms/step - loss: 0.7401 - accuracy: 0.7388 - val_loss: 1.7023 - val_accuracy: 0.5257
Epoch 16/20
56/56 [==============================] - 2s 39ms/step - loss: 0.7304 - accuracy: 0.7422 - val_loss: 1.5239 - val_accuracy: 0.5459
Epoch 17/20
56/56 [==============================] - 2s 39ms/step - loss: 0.6588 - accuracy: 0.7662 - val_loss: 1.6412 - val_accuracy: 0.5324
Epoch 18/20
56/56 [==============================] - 2s 39ms/step - loss: 0.5624 - accuracy: 0.8008 - val_loss: 1.7139 - val_accuracy: 0.5302
Epoch 19/20
56/56 [==============================] - 2s 39ms/step - loss: 0.5093 - accuracy: 0.8131 - val_loss: 1.6826 - val_accuracy: 0.5414
Epoch 20/20
56/56 [==============================] - 2s 39ms/step - loss: 0.4771 - accuracy: 0.8371 - val_loss: 1.9421 - val_accuracy: 0.5481
```

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling (Rescaling)       (None, 180, 180, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 180, 180, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 90, 90, 32)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 90, 90, 32)        0         
                                                                 
 conv2d_1 (Conv2D)           (None, 90, 90, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 45, 45, 64)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 45, 45, 64)        0         
                                                                 
 flatten (Flatten)           (None, 129600)            0         
                                                                 
 dense (Dense)               (None, 128)               16588928  
```

```
 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 9)                 1161

=================================================================
Total params: 16,609,481
Trainable params: 16,609,481
Non-trainable params: 0
_________________________________________________________________
```

```
#Visualizing training results
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()
```

Observations

- Training accuracy = 83%
- Validation accuracy = 54%
- It is not in par with the training accuracy.
- The validation loss as observed is very high.
- Indicative of some Overfit in the model.
- We could add some Dropout layers and remove the BatchNormalization layers.
- And by adding a few more layers, we could improve the accuracy by trying to extract more features.

Model 1

```
#Creating the Model
model_update=Sequential([
    tf.keras.layers.experimental.preprocessing.Rescaling(scale=1./255., offset=0.0,),

    Conv2D(32,(3,3),input_shape=(img_height,img_width,3),activation='relu',padding='same'),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.7),

    Conv2D(64,(3,3),activation='relu',padding='same'),
    Conv2D(64,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.7),

    Conv2D(128,(3,3),activation='relu',padding='same'),
    Conv2D(128,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.7),

    Flatten(),
    Dense(100, activation='relu'),
    Dropout(0.25),
    Dense(9, activation='softmax')
])


# Compiling the model
model_update.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics='accuracy')


# Training the model
epochs = 20
history = model_update.fit(
  train_ds,
  validation_data=val_ds,
```

```
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 8s 106ms/step - loss: 2.2740 - accuracy: 0.1775 - val_loss: 2.0637 - val_accuracy: 0.1790
Epoch 2/20
56/56 [==============================] - 5s 95ms/step - loss: 2.0616 - accuracy: 0.1953 - val_loss: 2.0375 - val_accuracy: 0.1924
Epoch 3/20
56/56 [==============================] - 5s 95ms/step - loss: 2.0291 - accuracy: 0.1853 - val_loss: 2.0087 - val_accuracy: 0.2081
Epoch 4/20
56/56 [==============================] - 5s 95ms/step - loss: 1.9824 - accuracy: 0.2104 - val_loss: 1.9304 - val_accuracy: 0.2438
Epoch 5/20
56/56 [==============================] - 5s 94ms/step - loss: 1.9372 - accuracy: 0.2383 - val_loss: 1.9062 - val_accuracy: 0.2148
Epoch 6/20
56/56 [==============================] - 5s 94ms/step - loss: 1.9155 - accuracy: 0.2706 - val_loss: 1.8836 - val_accuracy: 0.3177
Epoch 7/20
56/56 [==============================] - 5s 95ms/step - loss: 1.9197 - accuracy: 0.2690 - val_loss: 1.8527 - val_accuracy: 0.3244
Epoch 8/20
56/56 [==============================] - 5s 94ms/step - loss: 1.8630 - accuracy: 0.3114 - val_loss: 1.8319 - val_accuracy: 0.3423
Epoch 9/20
56/56 [==============================] - 5s 95ms/step - loss: 1.8413 - accuracy: 0.3131 - val_loss: 1.7947 - val_accuracy: 0.3333
Epoch 10/20
56/56 [==============================] - 5s 95ms/step - loss: 1.7644 - accuracy: 0.3510 - val_loss: 1.6951 - val_accuracy: 0.3937
Epoch 11/20
56/56 [==============================] - 5s 95ms/step - loss: 1.7268 - accuracy: 0.3650 - val_loss: 1.6955 - val_accuracy: 0.3803
Epoch 12/20
56/56 [==============================] - 5s 94ms/step - loss: 1.6781 - accuracy: 0.3945 - val_loss: 1.6847 - val_accuracy: 0.3982
Epoch 13/20
56/56 [==============================] - 5s 94ms/step - loss: 1.6446 - accuracy: 0.4001 - val_loss: 1.5873 - val_accuracy: 0.4407
Epoch 14/20
56/56 [==============================] - 5s 94ms/step - loss: 1.6996 - accuracy: 0.3761 - val_loss: 1.6433 - val_accuracy: 0.4094
Epoch 15/20
56/56 [==============================] - 5s 94ms/step - loss: 1.6396 - accuracy: 0.3884 - val_loss: 1.6273 - val_accuracy: 0.4318
Epoch 16/20
56/56 [==============================] - 5s 94ms/step - loss: 1.5995 - accuracy: 0.4079 - val_loss: 1.5883 - val_accuracy: 0.4273
Epoch 17/20
56/56 [==============================] - 5s 94ms/step - loss: 1.5951 - accuracy: 0.4169 - val_loss: 1.6448 - val_accuracy: 0.4027
Epoch 18/20
56/56 [==============================] - 5s 94ms/step - loss: 1.5703 - accuracy: 0.4202 - val_loss: 1.5952 - val_accuracy: 0.4161
Epoch 19/20
56/56 [==============================] - 5s 95ms/step - loss: 1.6341 - accuracy: 0.3968 - val_loss: 1.6965 - val_accuracy: 0.3826
Epoch 20/20
56/56 [==============================] - 5s 95ms/step - loss: 1.6100 - accuracy: 0.4152 - val_loss: 1.6206 - val_accuracy: 0.4273
```

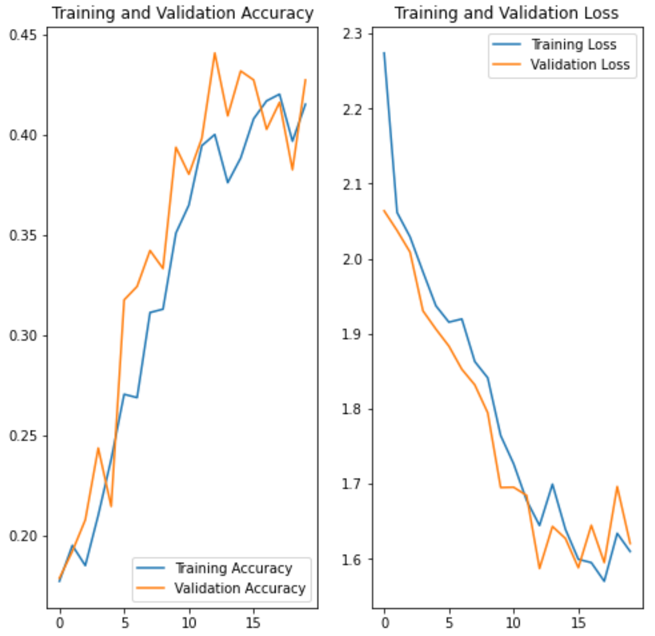
```
# Visualizing the results
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Observations

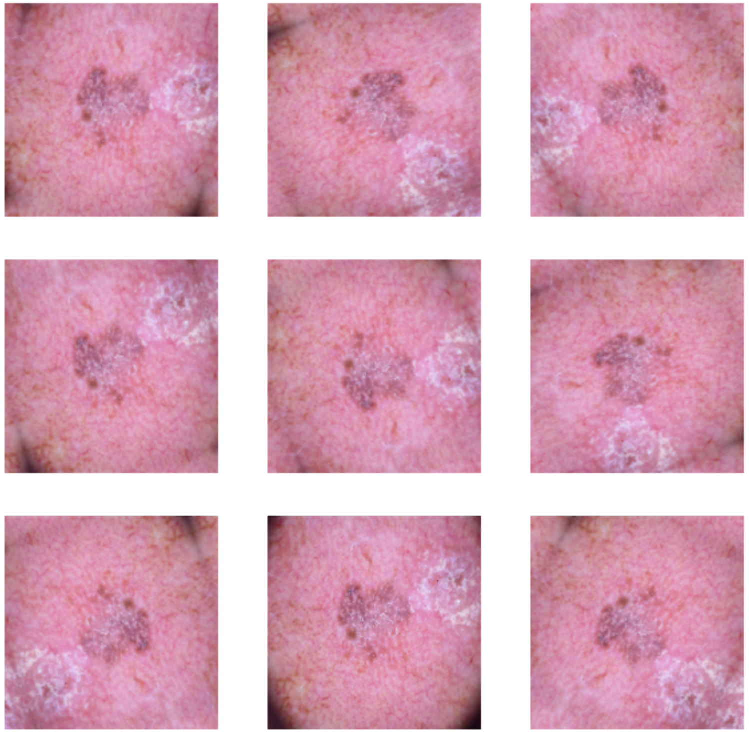
- Training Accuracy = 41%.
- Validation Accuracy = 43%.
- This is a much better model compared to the previous model as there seems to be No Overfit.

5. Data Augmentation

```
# Specifying the Augmentation
data_augmentation=tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2)
])
```

```
# Visualizing the Augmented Data
image, label = next(iter(train_ds))
image=np.array(image,np.int32)
```

```
plt.figure(figsize=(10, 10))
for i in range(9):
  augmented_image = data_augmentation(image)
  ax = plt.subplot(3, 3, i + 1)
  augmented_image1=np.array(augmented_image[0],np.int32)
  plt.imshow((augmented_image1))
  plt.axis("off")
```



6. Model 2

```
# Creating the Model
model_augmented=Sequential([
    tf.keras.layers.experimental.preprocessing.Rescaling(scale=1./255., offset=0.0,),

    data_augmentation,

    Conv2D(32,(3,3),input_shape=(img_height,img_width,3),activation='relu',padding='same'),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.7),

    Conv2D(64,(3,3),activation='relu',padding='same'),
    Conv2D(64,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
```

```
    Dropout(0.7),

    Conv2D(128,(3,3),activation='relu',padding='same'),
    Conv2D(128,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.7),

    Flatten(),
    Dense(100, activation='relu'),
    Dropout(0.25),
    Dense(9, activation='softmax')
])


# Compiling the model
model_augmented.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics='accuracy')


# Training the model
epochs = 20
history = model_augmented.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 8s 117ms/step - loss: 2.1188 - accuracy: 0.2015 - val_loss: 2.0356 - val_accuracy: 0.2058
Epoch 2/20
56/56 [==============================] - 6s 113ms/step - loss: 2.0418 - accuracy: 0.1903 - val_loss: 2.0270 - val_accuracy: 0.2058
Epoch 3/20
56/56 [==============================] - 6s 114ms/step - loss: 2.0364 - accuracy: 0.2059 - val_loss: 2.0284 - val_accuracy: 0.2058
Epoch 4/20
56/56 [==============================] - 6s 114ms/step - loss: 2.0387 - accuracy: 0.1881 - val_loss: 2.0213 - val_accuracy: 0.1924
Epoch 5/20
56/56 [==============================] - 6s 114ms/step - loss: 2.0388 - accuracy: 0.1964 - val_loss: 2.0196 - val_accuracy: 0.2058
Epoch 6/20
56/56 [==============================] - 6s 113ms/step - loss: 2.0082 - accuracy: 0.2160 - val_loss: 1.9944 - val_accuracy: 0.2506
Epoch 7/20
56/56 [==============================] - 6s 113ms/step - loss: 1.9295 - accuracy: 0.2773 - val_loss: 1.8574 - val_accuracy: 0.2886
Epoch 8/20
56/56 [==============================] - 6s 113ms/step - loss: 1.8633 - accuracy: 0.3013 - val_loss: 2.0173 - val_accuracy: 0.2282
Epoch 9/20
56/56 [==============================] - 6s 113ms/step - loss: 1.8106 - accuracy: 0.3315 - val_loss: 1.7155 - val_accuracy: 0.4206
Epoch 10/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6941 - accuracy: 0.3828 - val_loss: 1.5929 - val_accuracy: 0.4295
Epoch 11/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6373 - accuracy: 0.4085 - val_loss: 1.6780 - val_accuracy: 0.3826
Epoch 12/20
56/56 [==============================] - 6s 112ms/step - loss: 1.8413 - accuracy: 0.3008 - val_loss: 1.6074 - val_accuracy: 0.4094
Epoch 13/20
56/56 [==============================] - 6s 112ms/step - loss: 1.6902 - accuracy: 0.3929 - val_loss: 1.6000 - val_accuracy: 0.4295
Epoch 14/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6618 - accuracy: 0.3906 - val_loss: 1.6010 - val_accuracy: 0.4318
Epoch 15/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6181 - accuracy: 0.4012 - val_loss: 1.6272 - val_accuracy: 0.3982
Epoch 16/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6256 - accuracy: 0.4124 - val_loss: 1.6542 - val_accuracy: 0.3826
```

```
Epoch 17/20
56/56 [==============================] - 6s 113ms/step - loss: 1.5961 - accuracy: 0.4174 - val_loss: 1.5566 - val_accuracy: 0.4519
Epoch 18/20
56/56 [==============================] - 7s 124ms/step - loss: 1.5963 - accuracy: 0.4213 - val_loss: 1.5447 - val_accuracy: 0.4430
Epoch 19/20
56/56 [==============================] - 6s 114ms/step - loss: 1.5758 - accuracy: 0.4336 - val_loss: 1.5517 - val_accuracy: 0.4519
Epoch 20/20
56/56 [==============================] - 6s 112ms/step - loss: 1.5924 - accuracy: 0.4291 - val_loss: 1.5549 - val_accuracy: 0.4497
```

```
# Visualizing the results
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()
```

Training and Validation Accuracy Training and Validation Loss

Observations

- Training accuracy = 42%.
- Validation accuracy = 44%.
- This is a much better model compared to the previous two models as there seems to be No Overfit.
- Data Augmentation has improved the model performance.

7. Checking for Class Imbalance

```
for i in class_names:
    directory =train_path+'/'+i+'/'
    class_directory = pathlib.Path(directory)
    length=len(list(class_directory.glob('*.jpg')))
    print(f'{i} has {length} samples.')
```

```
actinic keratosis has 114 samples.
basal cell carcinoma has 376 samples.
dermatofibroma has 95 samples.
melanoma has 438 samples.
nevus has 357 samples.
pigmented benign keratosis has 462 samples.
seborrheic keratosis has 77 samples.
squamous cell carcinoma has 181 samples.
vascular lesion has 139 samples.
```

- The samples of various classes are not in equal proportion.
- There is a significant Class Imbalance observed.
- The class with the least number of samples is Seborrheic Keratosis with 77.
- The class that dominates the data in terms of proportionate number of samples is Pigmented Benign Keratosis with sample size of 462.

8. Using Augmentor for Class Imbalance Treatment

```
# Installing Augmentor
!pip install Augmentor
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting Augmentor
  Downloading Augmentor-0.2.10-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.8/dist-packages (from Augmentor) (1.21.6)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.8/dist-packages (from Augmentor) (7.1.2)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.8/dist-packages (from Augmentor) (4.64.1)
Requirement already satisfied: future>=0.16.0 in /usr/local/lib/python3.8/dist-packages (from Augmentor) (0.16.0)
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.10
```

```
# Using Augmentor
path_to_training_dataset=train_path
import Augmentor
for i in class_names:
    p = Augmentor.Pipeline(path_to_training_dataset + '/' + i)
    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 classes are sparse.
```

```
Initialised with 114 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/actinic keratosis/output.Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FE9
Initialised with 376 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/basal cell carcinoma/output.Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB s
Initialised with 95 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/dermatofibroma/output.Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FE9D8D
Initialised with 438 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/melanoma/output.Processing <PIL.Image.Image image mode=RGB size=2048x1536 at 0x7FE9D8E16D3
Initialised with 357 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/nevus/output.Processing <PIL.Image.Image image mode=RGB size=1504x1129 at 0x7FE9D8CD8A30>:
Initialised with 462 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/pigmented benign keratosis/output.Processing <PIL.Image.Image image mode=RGB size=600x450
Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/seborrheic keratosis/output.Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x
Initialised with 181 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/squamous cell carcinoma/output.Processing <PIL.Image.Image image mode=RGB size=600x450 at
Initialised with 139 image(s) found.
Output directory set to /content/gdrive/MyDrive/PG_AI_ML/TestData/skin-cancer-data/Train/vascular lesion/output.Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FE9D8
```

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

```
4500
```

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

```
lesion_list_new = [os.path.basename(os.path.dirname(os.path.dirname(y))) for y in glob(os.path.join(data_dir_train, '*','output', '*.jpg'))]
```

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

```
for i in class_names:
    directory =train_path+'/'+i+'/'
    directory_out =train_path+'/'+i+'/output/'
    class_directory = pathlib.Path(directory)
    class_directory_out = pathlib.Path(directory_out)
    length=len(list(class_directory.glob('*.jpg')))
    length_out=len(list(class_directory_out.glob('*.jpg')))
    length_tot=length+length_out
    print(f'{i} has {length_tot} samples.')
```

```
actinic keratosis has 614 samples.
basal cell carcinoma has 876 samples.
dermatofibroma has 595 samples.
melanoma has 938 samples.
nevus has 857 samples.
pigmented benign keratosis has 962 samples.
```

```
seborrheic keratosis has 577 samples.
squamous cell carcinoma has 681 samples.
vascular lesion has 639 samples.
```

Observations:

- The Augmentor has helped decrease the imbalance in class images and that can be viewed from above.

9. Modelling Augmented Data

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

```
# Creating the Train Data Set
data_dir_train=train_path
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123, label_mode='categorical',
  validation_split = 0.2,
  subset = 'training',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

```
# Creating the Validation Data Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123, label_mode='categorical',
  validation_split = 0.2,
  subset = 'validation',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

10. Model 3

```
# Creating the Model
model_final=Sequential([
    tf.keras.layers.experimental.preprocessing.Rescaling(scale=1./255., offset=0.0,),

    Conv2D(32,(3,3),input_shape=(img_height,img_width,3),activation='relu',padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.1),
```

```
    Conv2D(64,(3,3),activation='relu',padding='same'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.1),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(9, activation='softmax')
])


# Compiling the Model
model_final.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics='accuracy')


# Training the Model
epochs = 30
## Your code goes here, use 50 epochs.
history = model_final.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/30
169/169 [==============================] - 34s 193ms/step - loss: 2.3669 - accuracy: 0.1534 - val_loss: 2.0330 - val_accuracy: 0.2829
Epoch 2/30
169/169 [==============================] - 30s 172ms/step - loss: 1.8049 - accuracy: 0.3197 - val_loss: 1.6019 - val_accuracy: 0.4254
Epoch 3/30
169/169 [==============================] - 30s 174ms/step - loss: 1.5430 - accuracy: 0.4125 - val_loss: 1.3691 - val_accuracy: 0.4922
Epoch 4/30
169/169 [==============================] - 30s 174ms/step - loss: 1.4124 - accuracy: 0.4594 - val_loss: 1.3728 - val_accuracy: 0.4662
Epoch 5/30
169/169 [==============================] - 32s 184ms/step - loss: 1.2963 - accuracy: 0.5106 - val_loss: 1.1074 - val_accuracy: 0.5702
Epoch 6/30
169/169 [==============================] - 30s 174ms/step - loss: 1.2005 - accuracy: 0.5371 - val_loss: 1.0794 - val_accuracy: 0.6065
Epoch 7/30
169/169 [==============================] - 32s 183ms/step - loss: 1.0756 - accuracy: 0.5909 - val_loss: 1.1127 - val_accuracy: 0.6125
Epoch 8/30
169/169 [==============================] - 30s 174ms/step - loss: 0.9857 - accuracy: 0.6332 - val_loss: 0.9622 - val_accuracy: 0.6600
Epoch 9/30
169/169 [==============================] - 30s 174ms/step - loss: 0.8941 - accuracy: 0.6647 - val_loss: 0.9345 - val_accuracy: 0.6540
Epoch 10/30
169/169 [==============================] - 30s 174ms/step - loss: 0.8196 - accuracy: 0.6923 - val_loss: 0.8159 - val_accuracy: 0.7120
Epoch 11/30
169/169 [==============================] - 30s 175ms/step - loss: 0.7120 - accuracy: 0.7283 - val_loss: 0.8834 - val_accuracy: 0.6897
Epoch 12/30
169/169 [==============================] - 32s 183ms/step - loss: 0.6875 - accuracy: 0.7483 - val_loss: 0.8149 - val_accuracy: 0.7350
Epoch 13/30
169/169 [==============================] - 31s 176ms/step - loss: 0.6050 - accuracy: 0.7693 - val_loss: 0.9540 - val_accuracy: 0.6912
Epoch 14/30
169/169 [==============================] - 30s 175ms/step - loss: 0.5415 - accuracy: 0.7945 - val_loss: 0.7751 - val_accuracy: 0.7543
Epoch 15/30
169/169 [==============================] - 30s 174ms/step - loss: 0.5116 - accuracy: 0.8036 - val_loss: 0.7766 - val_accuracy: 0.7342
Epoch 16/30
169/169 [==============================] - 30s 174ms/step - loss: 0.4909 - accuracy: 0.8053 - val_loss: 0.7293 - val_accuracy: 0.7840
Epoch 17/30
169/169 [==============================] - 30s 173ms/step - loss: 0.4585 - accuracy: 0.8194 - val_loss: 0.7919 - val_accuracy: 0.7506
```

```
Epoch 18/30
169/169 [==============================] - 32s 183ms/step - loss: 0.4227 - accuracy: 0.8338 - val_loss: 0.7849 - val_accuracy: 0.7572
Epoch 19/30
169/169 [==============================] - 30s 174ms/step - loss: 0.4264 - accuracy: 0.8325 - val_loss: 0.7737 - val_accuracy: 0.7751
Epoch 20/30
169/169 [==============================] - 30s 175ms/step - loss: 0.3707 - accuracy: 0.8553 - val_loss: 0.7189 - val_accuracy: 0.7869
Epoch 21/30
169/169 [==============================] - 30s 174ms/step - loss: 0.3796 - accuracy: 0.8539 - val_loss: 0.7893 - val_accuracy: 0.7832
Epoch 22/30
169/169 [==============================] - 30s 173ms/step - loss: 0.3669 - accuracy: 0.8626 - val_loss: 0.8566 - val_accuracy: 0.7721
Epoch 23/30
169/169 [==============================] - 30s 174ms/step - loss: 0.3470 - accuracy: 0.8678 - val_loss: 0.8063 - val_accuracy: 0.7966
Epoch 24/30
169/169 [==============================] - 32s 183ms/step - loss: 0.3309 - accuracy: 0.8646 - val_loss: 0.8482 - val_accuracy: 0.7966
Epoch 25/30
169/169 [==============================] - 30s 174ms/step - loss: 0.3099 - accuracy: 0.8793 - val_loss: 0.8844 - val_accuracy: 0.7795
Epoch 26/30
169/169 [==============================] - 30s 173ms/step - loss: 0.2975 - accuracy: 0.8806 - val_loss: 0.9592 - val_accuracy: 0.7632
Epoch 27/30
169/169 [==============================] - 30s 174ms/step - loss: 0.2926 - accuracy: 0.8848 - val_loss: 0.8669 - val_accuracy: 0.7944
Epoch 28/30
169/169 [==============================] - 30s 174ms/step - loss: 0.2799 - accuracy: 0.8898 - val_loss: 0.9213 - val_accuracy: 0.7706
Epoch 29/30
169/169 [==============================] - 30s 174ms/step - loss: 0.2870 - accuracy: 0.8898 - val_loss: 0.9054 - val_accuracy: 0.7877
```

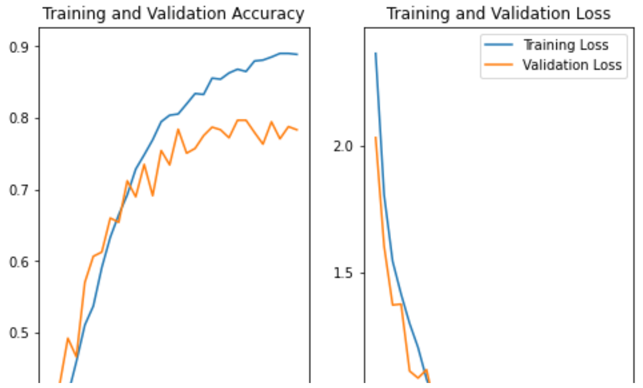
```
#Visualizing the model results
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Observations

- Training accuracy = ~88%.
- Validation accuracy = ~78%.
- Though the model accuracy has improved, the class rebalance has helped treat the overfitting to some extent.
- Much better models could be built or tried out using more epochs and more layers.

✓ 0s completed at 12:41 AM