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

Importing all the important 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
```

```
## If you are using the data by mounting the google drive, use the following :
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).
```

```
root_dir = "/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration"
```

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.

```
# Defining the path for train and test images
data_dir_train = pathlib.Path(root_dir + "/Train")
data_dir_test = pathlib.Path(root_dir + "/Test")
```

```
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:

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

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

```
## Write your train dataset here
## Note use seed=123 while creating your dataset using tf.keras.preprocessing.image_dataset_from_directory
## Note, make sure your resize your images to the size img_height*img_width, while writting the dataset
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    seed=123,
```

```
    validation_split=0.2,
    subset='training',
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

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

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

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

```
# 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)
```

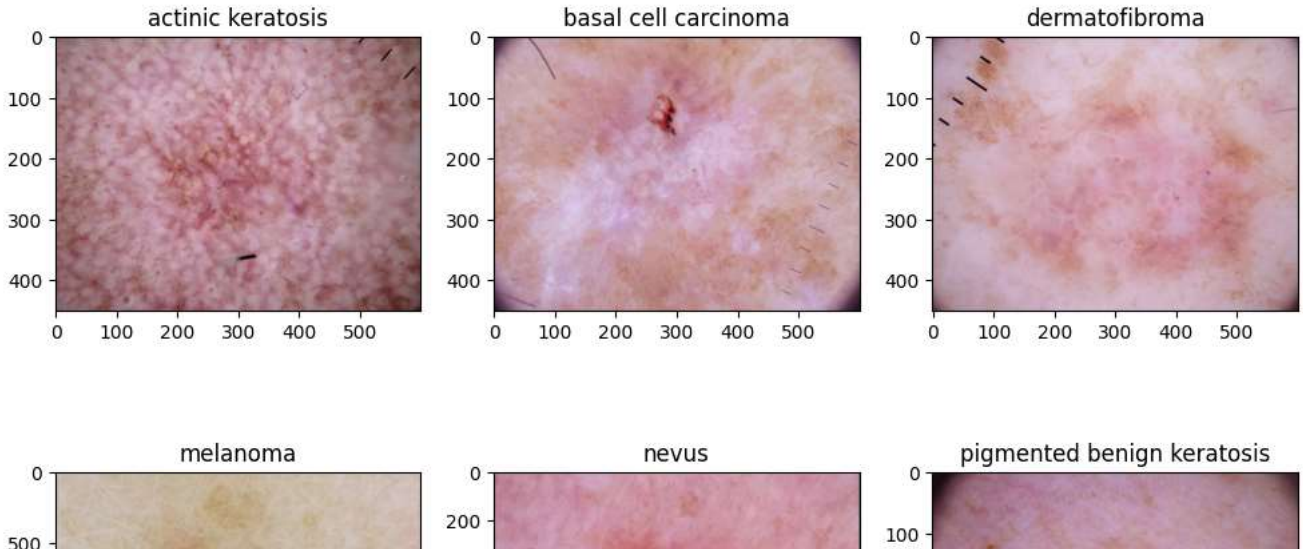
```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', 'nevus', 'pigmented benign keratosis', 'seborrheic keratosis
```

Visualize the data

Code to visualize one instance of all the nine classes present in the dataset

```
import matplotlib.pyplot as plt

### your code goes here, you can use training or validation data to visualize
plt.figure(figsize=(12,12))
for i in range(len(class_names)):
    plt.subplot(3,3,i+1)
    imgValue = plt.imread(str(list(data_dir_train.glob(class_names[i]+'/*.jpg'))[1]))
    plt.title(class_names[i])
    plt.imshow(imgValue)
```



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.

```
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 first model

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]`

```
### Your code goes here
from keras.layers import Conv2D,Dense,Dropout,Flatten,MaxPool2D
classes_num = 9
model_1 = Sequential([
    layers.experimental.preprocessing.Rescaling(scale=1./255,input_shape=(img_height,img_width,3))
])
# Creating Convolution layer with 64 features, 3x3 filter and activation function as relu  with 2x2 pooling
model_1.add(layers.Conv2D(64,(3,3),padding = 'same',activation='relu'))
model_1.add(layers.MaxPooling2D())

# Creating Convolution layer with 128 features, 3x3 filter and activation function as relu with 2x2 pooling
model_1.add(layers.Conv2D(128,(3,3),padding = 'same',activation='relu'))
model_1.add(layers.MaxPooling2D())

model_1.add(Flatten())

model_1.add(Dense(256,activation='relu'))#added
model_1.add(Dense(classes_num,activation='softmax'))
```

Compile the model

Choose an appropirate optimiser and loss function for model training

```
### Todo, choose an appropirate optimiser and loss function
model_1.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

```
# View the summary of all layers
model_1.summary()
```

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

 conv2d (Conv2D)             (None, 180, 180, 64)      1792

 max_pooling2d (MaxPooling2  (None, 90, 90, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 90, 90, 128)       73856

 max_pooling2d_1 (MaxPoolin  (None, 45, 45, 128)       0
 g2D)

 flatten (Flatten)           (None, 259200)            0

 dense (Dense)               (None, 256)               66355456

 dense_1 (Dense)             (None, 9)                 2313

=================================================================
Total params: 66433417 (253.42 MB)
Trainable params: 66433417 (253.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train the model

```
epochs = 20
history = model_1.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 190s 2s/step - loss: 2.9365 - accuracy: 0.2031 - val_loss: 1.9519 - val_accuracy: 0.3289
Epoch 2/20
56/56 [==============================] - 4s 71ms/step - loss: 1.7837 - accuracy: 0.3298 - val_loss: 1.7806 - val_accuracy: 0.3870
Epoch 3/20
56/56 [==============================] - 4s 67ms/step - loss: 1.5690 - accuracy: 0.4414 - val_loss: 1.5050 - val_accuracy: 0.5011
Epoch 4/20
56/56 [==============================] - 4s 67ms/step - loss: 1.4457 - accuracy: 0.5017 - val_loss: 1.7488 - val_accuracy: 0.4027
Epoch 5/20
56/56 [==============================] - 4s 70ms/step - loss: 1.3494 - accuracy: 0.5368 - val_loss: 1.4166 - val_accuracy: 0.5213
Epoch 6/20
56/56 [==============================] - 4s 67ms/step - loss: 1.1975 - accuracy: 0.5798 - val_loss: 1.4673 - val_accuracy: 0.5391
Epoch 7/20
56/56 [==============================] - 4s 68ms/step - loss: 1.0640 - accuracy: 0.6256 - val_loss: 1.5961 - val_accuracy: 0.4944
Epoch 8/20
56/56 [==============================] - 4s 69ms/step - loss: 0.9509 - accuracy: 0.6602 - val_loss: 1.5026 - val_accuracy: 0.5145
Epoch 9/20
56/56 [==============================] - 4s 69ms/step - loss: 0.8311 - accuracy: 0.7093 - val_loss: 1.5920 - val_accuracy: 0.5145
Epoch 10/20
56/56 [==============================] - 4s 68ms/step - loss: 0.7282 - accuracy: 0.7511 - val_loss: 1.9517 - val_accuracy: 0.4966
Epoch 11/20
56/56 [==============================] - 4s 70ms/step - loss: 0.6393 - accuracy: 0.7690 - val_loss: 1.8697 - val_accuracy: 0.5123
Epoch 12/20
56/56 [==============================] - 4s 69ms/step - loss: 0.5598 - accuracy: 0.8064 - val_loss: 1.9599 - val_accuracy: 0.5235
Epoch 13/20
56/56 [==============================] - 4s 68ms/step - loss: 0.4936 - accuracy: 0.8270 - val_loss: 2.2110 - val_accuracy: 0.4765
Epoch 14/20
56/56 [==============================] - 4s 69ms/step - loss: 0.6459 - accuracy: 0.7930 - val_loss: 1.9202 - val_accuracy: 0.4966
Epoch 15/20
56/56 [==============================] - 4s 68ms/step - loss: 0.4681 - accuracy: 0.8259 - val_loss: 2.1893 - val_accuracy: 0.5213
Epoch 16/20
56/56 [==============================] - 4s 68ms/step - loss: 0.4046 - accuracy: 0.8577 - val_loss: 2.4290 - val_accuracy: 0.5436
Epoch 17/20
56/56 [==============================] - 4s 70ms/step - loss: 0.3570 - accuracy: 0.8767 - val_loss: 2.3873 - val_accuracy: 0.5280
Epoch 18/20
56/56 [==============================] - 4s 69ms/step - loss: 0.2799 - accuracy: 0.8979 - val_loss: 2.2204 - val_accuracy: 0.5056
Epoch 19/20
56/56 [==============================] - 4s 67ms/step - loss: 0.2694 - accuracy: 0.8984 - val_loss: 2.4624 - val_accuracy: 0.5190
Epoch 20/20
56/56 [==============================] - 4s 67ms/step - loss: 0.2516 - accuracy: 0.8940 - val_loss: 2.7137 - val_accuracy: 0.5078
```

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()
```

Findings from First model:--

We can infer from the above graph that

1. The validation accuracy stall at ~51% accuracy in training process, where as the training accuracy increases over time .
2. The training loss dereases with each epochs while the validation loss increases.
3. Training accuracy and validation accuracy are off by a large margin, as model has achieved training accuracy of **~89%** while it has achieved around **51%** accuracy on the validation set.
4. The difference between training and validation accuracy is **very huge** which in turn is a sign of **overfitting**.

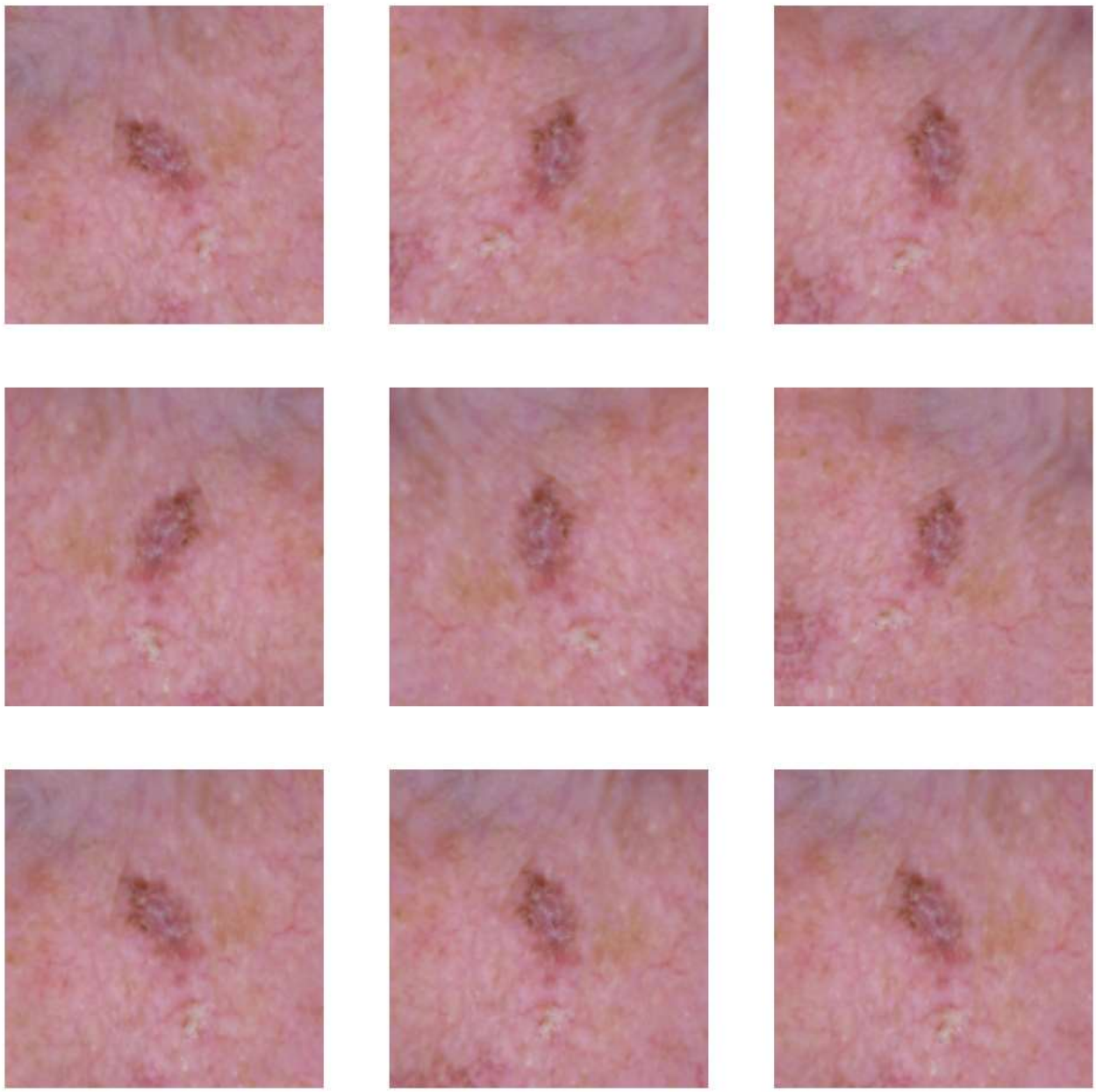
```
# We observed that our model is overfitting. To handle this, We can introduce more sample by augmenting the exisiting images.
```

```
# We observed that our model is overfitting. To handle this, we can introduce more sample by augmenting the existing images.
# Augmentation Strategies chosen
# 1. Random Flip around horizontal axis.
# 2. Random Rotation
# 3. Random Zoom

data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",input_shape=(img_height,img_width,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)


# Visualizing the augmentation strategy

plt.figure(figsize=(12, 12))
for images, _ in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    augmented_img = data_augmentation(images)
    plt.imshow(augmented_img[0].numpy().astype("uint8"))
    plt.axis("off")
```



Create the second model with the augmentation strategy.

Create the model, compile and train the model

```
## We have used Dropout layer since there is an evidence of overfitting in your findings

model_2 = Sequential([
    data_augmentation,
    layers.experimental.preprocessing.Rescaling(scale=1./255,input_shape=(img_height,img_width,3))
])
# Creating Convolution layer with 64 features, 3x3 filter and activation function as relu  with 2x2 pooling
model_2.add(layers.Conv2D(64,(3,3),padding = 'same',activation='relu'))
model_2.add(layers.MaxPooling2D())

# Creating Convolution layer with 128 features, 3x3 filter and activation function as relu with 2x2 pooling
model_2.add(layers.Conv2D(128,(3,3),padding = 'same',activation='relu'))
model_2.add(layers.MaxPooling2D())
# Now this time adding a 20% dropout after the convolution layers
model_2.add(layers.Dropout(0.2))

model_2.add(Flatten())

model_2.add(Dense(256,activation='relu'))#added
model_2.add(Dense(classes_num,activation='softmax'))
```

Compiling the model

```
# compiling the model
model_2.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
# Showing the Summary of all the layer in model_2
model_2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_2 (Conv2D)           (None, 180, 180, 64)      1792

 max_pooling2d_2 (MaxPoolin  (None, 90, 90, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 90, 90, 128)       73856

 max_pooling2d_3 (MaxPoolin  (None, 45, 45, 128)       0
 g2D)

 dropout (Dropout)           (None, 45, 45, 128)       0

 flatten_1 (Flatten)         (None, 259200)            0

 dense_2 (Dense)             (None, 256)               66355456

 dense_3 (Dense)             (None, 9)                 2313

=================================================================
Total params: 66433417 (253.42 MB)
Trainable params: 66433417 (253.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training the model

```
## Training the model for 20 epochs
epochs = 20
history = model_2.fit(
  train_ds,
  validation_data=val_ds,
```

```
  epochs=epochs
)
```

```
Epoch 1/20
56/56 [==============================] - 8s 114ms/step - loss: 2.9331 - accuracy: 0.2556 - val_loss: 1.8843 - val_accuracy: 0.2841
Epoch 2/20
56/56 [==============================] - 6s 112ms/step - loss: 1.7557 - accuracy: 0.3627 - val_loss: 1.6295 - val_accuracy: 0.4609
Epoch 3/20
56/56 [==============================] - 6s 113ms/step - loss: 1.6004 - accuracy: 0.4397 - val_loss: 1.5084 - val_accuracy: 0.4698
Epoch 4/20
56/56 [==============================] - 6s 111ms/step - loss: 1.5033 - accuracy: 0.4771 - val_loss: 1.5175 - val_accuracy: 0.4989
Epoch 5/20
56/56 [==============================] - 6s 110ms/step - loss: 1.4171 - accuracy: 0.5061 - val_loss: 1.4064 - val_accuracy: 0.5324
Epoch 6/20
56/56 [==============================] - 6s 110ms/step - loss: 1.4123 - accuracy: 0.4983 - val_loss: 1.4995 - val_accuracy: 0.5101
Epoch 7/20
56/56 [==============================] - 6s 112ms/step - loss: 1.3662 - accuracy: 0.5240 - val_loss: 1.5162 - val_accuracy: 0.4877
Epoch 8/20
56/56 [==============================] - 6s 108ms/step - loss: 1.3833 - accuracy: 0.5095 - val_loss: 1.4214 - val_accuracy: 0.5280
Epoch 9/20
56/56 [==============================] - 6s 109ms/step - loss: 1.2919 - accuracy: 0.5435 - val_loss: 1.3871 - val_accuracy: 0.4989
Epoch 10/20
56/56 [==============================] - 6s 108ms/step - loss: 1.2581 - accuracy: 0.5502 - val_loss: 1.4637 - val_accuracy: 0.4922
Epoch 11/20
56/56 [==============================] - 6s 110ms/step - loss: 1.2937 - accuracy: 0.5413 - val_loss: 1.4207 - val_accuracy: 0.5034
Epoch 12/20
56/56 [==============================] - 6s 107ms/step - loss: 1.2507 - accuracy: 0.5547 - val_loss: 1.3025 - val_accuracy: 0.5459
Epoch 13/20
56/56 [==============================] - 6s 110ms/step - loss: 1.1997 - accuracy: 0.5636 - val_loss: 1.3344 - val_accuracy: 0.5369
Epoch 14/20
56/56 [==============================] - 6s 108ms/step - loss: 1.1408 - accuracy: 0.6049 - val_loss: 1.3481 - val_accuracy: 0.5615
Epoch 15/20
56/56 [==============================] - 6s 110ms/step - loss: 1.1678 - accuracy: 0.5725 - val_loss: 1.4005 - val_accuracy: 0.5593
Epoch 16/20
56/56 [==============================] - 6s 107ms/step - loss: 1.1398 - accuracy: 0.5826 - val_loss: 1.3637 - val_accuracy: 0.5391
Epoch 17/20
56/56 [==============================] - 6s 110ms/step - loss: 1.1137 - accuracy: 0.6083 - val_loss: 1.3289 - val_accuracy: 0.5503
Epoch 18/20
56/56 [==============================] - 6s 110ms/step - loss: 1.1124 - accuracy: 0.5960 - val_loss: 1.6059 - val_accuracy: 0.4452
Epoch 19/20
56/56 [==============================] - 6s 108ms/step - loss: 1.1102 - accuracy: 0.5915 - val_loss: 1.3583 - val_accuracy: 0.5190
Epoch 20/20
56/56 [==============================] - 6s 107ms/step - loss: 1.1059 - accuracy: 0.5993 - val_loss: 1.4286 - val_accuracy: 0.5503
```

Visualizing the results

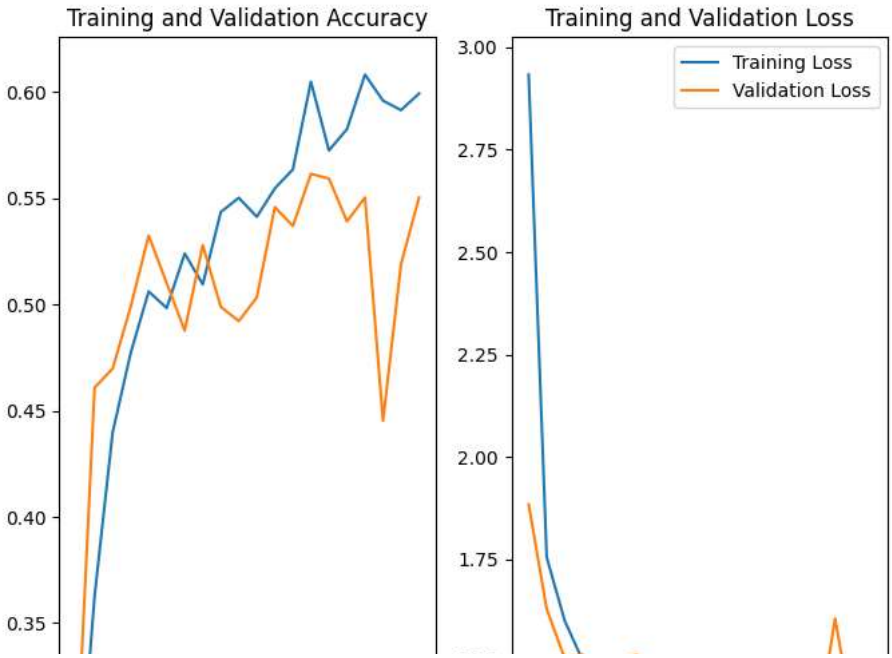
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Findings from Second model:--

We can infer from above graph that

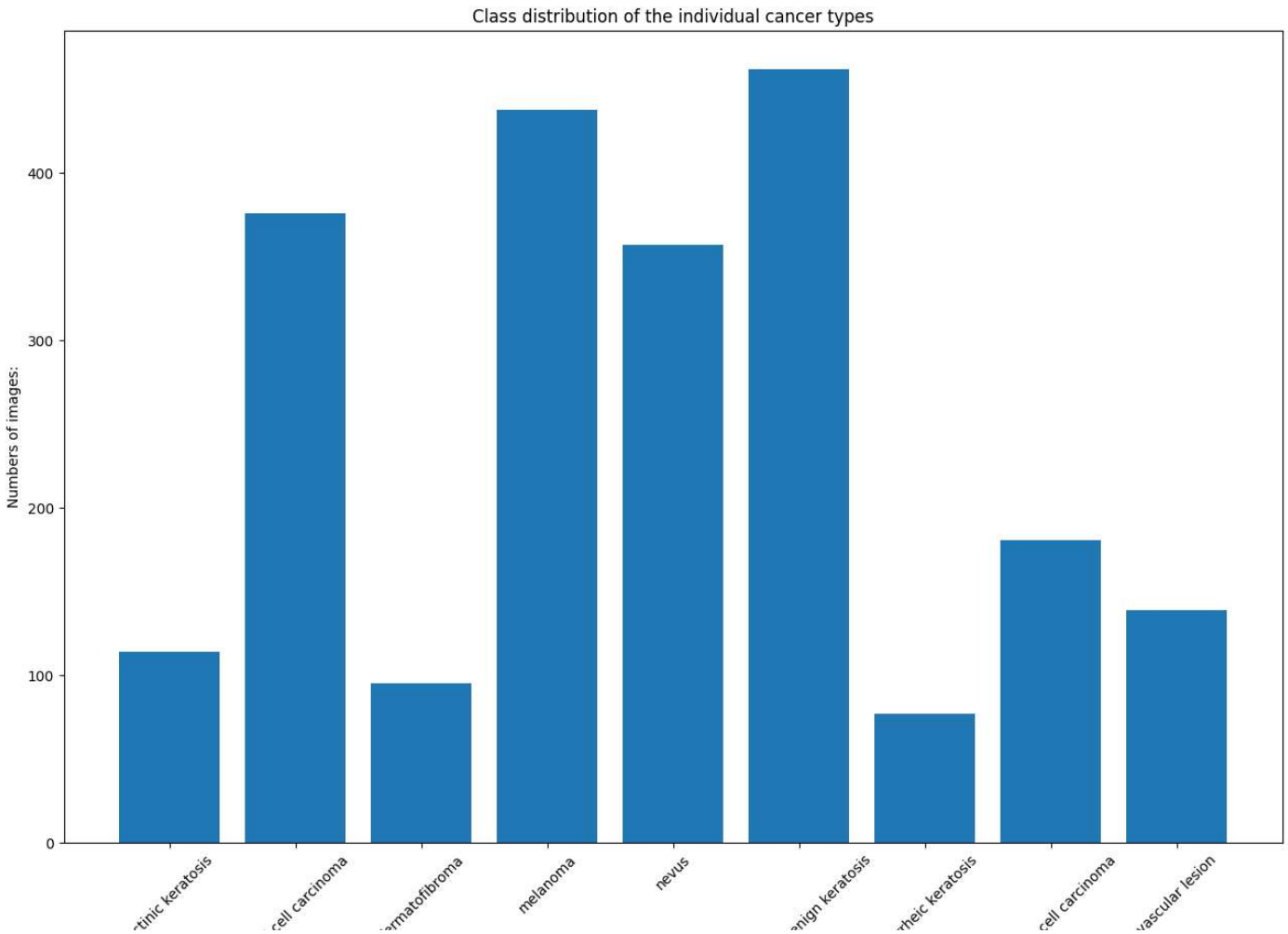
1. The training accuracy increases over time, where as the validation accuracy increases and stall at 55% accuracy during training process.
2. **Both training loss as well as validation loss decreases during training process with each epochs.**
3. The gap between training accuracy and validation accuracy have decreased from previous model, and it has achieved around **55%** accuracy on the validation set.
4. The difference in accuracy between training and validation is **very less**, which is good as it signifies that model has not overfit.

Find the distribution of classes in the training dataset.

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.

```
# Plot the images to check if all the cancer types are equally distributed
fig = plt.figure(figsize=(12,8))
ax = fig.add_axes([0,0,1,1])
x=[]
y=[]
for i in range(len(class_names)):
  x.append(class_names[i])
  y.append(len(list(data_dir_train.glob(class_names[i]+'/*.jpg'))))

ax.bar(x,y)
ax.set_ylabel('Numbers of images:')
ax.set_title('Class distribution of the individual cancer types')
plt.xticks(rotation=45)
plt.show()
```




```
## Your code goes here.
for indx in range(len(class_names)):
  print(class_names[indx],' - ',len(list(data_dir_train.glob(class_names[indx]+'/*.jpg'))))
```

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

Findings

1.) Which class has the least number of samples?

Ans. 1) **seborrheic keratosis** has the least number of samples which is 77 in total.

2.) Which classes dominate the data in terms proportionate number of samples?

Ans. 2) **pigmented benign keratosis** dominates with 462 samples in total.

Rectify the class imbalance

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

```
!pip install Augmentor
```

```
Collecting Augmentor
  Downloading Augmentor-0.2.12-py2.py3-none-any.whl (38 kB)
```

```
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (9.4.0)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (4.66.1)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from Augmentor) (1.23.5)
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.12
```

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.

```
path_to_training_dataset= root_dir + "/Train/"
import Augmentor
for i in class_names:
    print(path_to_training_dataset + i)
    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.
```

```
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/actinic keratosis
Initialised with 114 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/actinic keratosis/ou
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma
Initialised with 376 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/dermatofibroma
Initialised with 95 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/dermatofibroma/outpu
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/melanoma
Initialised with 438 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/melanoma/output.Proc
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/nevus
Initialised with 357 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/nevus/output.Process
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/pigmented benign keratosis
Initialised with 462 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/pigmented benign ker
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/seborrheic keratosis
Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/seborrheic keratosis
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous cell carcinoma
Initialised with 181 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous cell carcir
/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular lesion
Initialised with 139 image(s) found.
Output directory set to /content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular lesion/outp
```

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
```

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

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

```
carcinoma_original_ISIC_0026324.jpg_1dad1662-562b-4f9e-9cdf-8c32936ae5bc.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0025301.jpg_9801df45-5447-435e-b010-d7b680ada005.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0030349.jpg_c4e11a26-194b-4355-987a-804131564149.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0030574.jpg_0487a266-c1ab-41b5-a142-892961c447b5.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0031266.jpg_a303a4a6-e635-4d60-9818-6db704aff223.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0025630.jpg_cafaea40-e0ea-46a7-8aa0-10dc9994aa5c.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0028542.jpg_c9ef2ed6-51a0-4b72-80e8-935c7b7da814.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0030526.jpg_c9d12f7c-d61a-43db-8fe2-0247b29dbb65.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0028693.jpg_c0add9ee-1418-4416-b03c-73ee46ed90ed.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0029352.jpg_08b583f5-2c6f-4173-bcf1-9c01138b5bac.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0024572.jpg_5be68fef-83c3-498b-99a3-7b03c751eef1.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0027004.jpg_d1d249f7-cda6-4b19-b940-05c3817d452f.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0024743.jpg_72be5ca1-b94c-41ec-b7eb-cdbea66b86f3.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0025576.jpg_d19e8716-b942-4f46-a3f7-86e057150b26.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0031552.jpg_cd330f34-fca9-46fe-b7f2-19372141feee.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0026865.jpg_a5ace0f0-5ebe-4ba3-977e-9ee4e6d09c4f.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0028147.jpg_454c536c-7401-4ef6-8874-413e072cf51f.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0027093.jpg_20f0e18e-d181-458c-986f-af3d92d6fb9f.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0031007.jpg_7c0eadbe-a709-4497-8c9b-c4182ed8b0d8.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0025046.jpg_65268b84-9c21-4c0d-9300-0656bf5dafcc.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0031284.jpg_a81d87d6-702d-4505-90ff-7bbbd58cd5f4.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0027120.jpg_cf0aa8ad-6c7f-4b79-8421-f02387787600.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0027759.jpg_f50cda79-eaca-45fd-971a-377cd7e503c1.jpg',
 '/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output/basal cell
carcinoma_original_ISIC_0030271.jpg_67c29fbc-abc9-4398-a88a-e14db53f8ea2.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'))]
lesion_list_new
```

```
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
      'basal cell carcinoma',
```

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

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

```
df2['Label'].value_counts()
```

```
    actinic keratosis             500
    basal cell carcinoma          500
    dermatofibroma                500
    melanoma                      500
    nevus                         500
    pigmented benign keratosis    500
    seborrheic keratosis          500
    squamous cell carcinoma       500
    vascular lesion               500
    Name: Label, dtype: int64
```

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

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

Create a training dataset

```
#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,
  seed=123,
  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.
```

Create a validation dataset

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  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.
```

Creating the model

```
## your code goes here
model_3 = Sequential([
    data_augmentation,
    layers.experimental.preprocessing.Rescaling(scale=1./255,input_shape=(img_height,img_width,3))
])
# Creating Convolution layer with 64 features, 3x3 filter and activation function as relu  with 2x2 pooling
model_3.add(layers.Conv2D(64,(3,3),padding = 'same',activation='relu'))
model_3.add(layers.MaxPooling2D())

# Creating Convolution layer with 128 features, 3x3 filter and activation function as relu with 2x2 pooling
model_3.add(layers.Conv2D(128,(3,3),padding = 'same',activation='relu'))
model_3.add(layers.MaxPooling2D())
# Now this time adding a 20% dropout after the convolution layers
model_3.add(layers.Dropout(0.2))

model_3.add(Flatten())

model_3.add(Dense(256,activation='relu'))#added
model_3.add(Dense(classes_num,activation='softmax'))
```

Compiling your model

```
# compiling the model
model_3.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
# Showing the Summary of all the layer in model_2
model_3.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_4 (Conv2D)           (None, 180, 180, 64)      1792

 max_pooling2d_4 (MaxPoolin  (None, 90, 90, 64)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 90, 90, 128)       73856

 max_pooling2d_5 (MaxPoolin  (None, 45, 45, 128)       0
 g2D)

 dropout_1 (Dropout)         (None, 45, 45, 128)       0

 flatten_2 (Flatten)         (None, 259200)            0

 dense_4 (Dense)             (None, 256)               66355456

 dense_5 (Dense)             (None, 9)                 2313

=================================================================
Total params: 66433417 (253.42 MB)
Trainable params: 66433417 (253.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train your model

```
epochs = 30
history = model_3.fit(
```

```
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/30
169/169 [==============================] - 56s 297ms/step - loss: 2.3180 - accuracy: 0.2819 - val_loss: 1.6956 - val_accuracy: 0.3571
Epoch 2/30
169/169 [==============================] - 39s 228ms/step - loss: 1.5241 - accuracy: 0.4212 - val_loss: 1.4274 - val_accuracy: 0.4566
Epoch 3/30
169/169 [==============================] - 41s 237ms/step - loss: 1.4395 - accuracy: 0.4579 - val_loss: 1.4312 - val_accuracy: 0.4647
Epoch 4/30
169/169 [==============================] - 42s 245ms/step - loss: 1.3661 - accuracy: 0.4777 - val_loss: 1.3053 - val_accuracy: 0.4981
Epoch 5/30
169/169 [==============================] - 42s 244ms/step - loss: 1.2942 - accuracy: 0.5141 - val_loss: 1.2148 - val_accuracy: 0.5397
Epoch 6/30
169/169 [==============================] - 42s 243ms/step - loss: 1.2366 - accuracy: 0.5258 - val_loss: 1.1963 - val_accuracy: 0.5449
Epoch 7/30
169/169 [==============================] - 40s 229ms/step - loss: 1.1799 - accuracy: 0.5525 - val_loss: 1.0737 - val_accuracy: 0.5798
Epoch 8/30
169/169 [==============================] - 42s 244ms/step - loss: 1.1248 - accuracy: 0.5773 - val_loss: 1.0748 - val_accuracy: 0.6050
Epoch 9/30
169/169 [==============================] - 38s 220ms/step - loss: 1.0734 - accuracy: 0.6024 - val_loss: 1.1901 - val_accuracy: 0.5338
Epoch 10/30
169/169 [==============================] - 38s 213ms/step - loss: 1.0289 - accuracy: 0.6066 - val_loss: 1.0112 - val_accuracy: 0.6206
Epoch 11/30
169/169 [==============================] - 39s 222ms/step - loss: 0.9847 - accuracy: 0.6324 - val_loss: 0.9370 - val_accuracy: 0.6533
Epoch 12/30
169/169 [==============================] - 39s 224ms/step - loss: 0.9564 - accuracy: 0.6393 - val_loss: 1.0436 - val_accuracy: 0.5909
Epoch 13/30
169/169 [==============================] - 42s 240ms/step - loss: 0.9317 - accuracy: 0.6463 - val_loss: 1.1125 - val_accuracy: 0.5791
Epoch 14/30
169/169 [==============================] - 41s 239ms/step - loss: 0.9107 - accuracy: 0.6552 - val_loss: 0.8997 - val_accuracy: 0.6526
Epoch 15/30
169/169 [==============================] - 41s 240ms/step - loss: 0.8834 - accuracy: 0.6699 - val_loss: 0.9295 - val_accuracy: 0.6466
Epoch 16/30
169/169 [==============================] - 41s 239ms/step - loss: 0.8312 - accuracy: 0.6871 - val_loss: 0.8689 - val_accuracy: 0.6793
Epoch 17/30
169/169 [==============================] - 42s 241ms/step - loss: 0.7825 - accuracy: 0.7092 - val_loss: 0.8663 - val_accuracy: 0.6771
Epoch 18/30
169/169 [==============================] - 41s 239ms/step - loss: 0.7560 - accuracy: 0.7209 - val_loss: 0.8093 - val_accuracy: 0.6956
Epoch 19/30
169/169 [==============================] - 38s 221ms/step - loss: 0.7499 - accuracy: 0.7205 - val_loss: 0.7832 - val_accuracy: 0.6964
Epoch 20/30
169/169 [==============================] - 38s 220ms/step - loss: 0.7490 - accuracy: 0.7266 - val_loss: 0.7088 - val_accuracy: 0.7209
Epoch 21/30
169/169 [==============================] - 42s 241ms/step - loss: 0.6813 - accuracy: 0.7452 - val_loss: 0.7187 - val_accuracy: 0.7134
Epoch 22/30
169/169 [==============================] - 41s 238ms/step - loss: 0.7147 - accuracy: 0.7316 - val_loss: 0.8029 - val_accuracy: 0.7142
Epoch 23/30
169/169 [==============================] - 41s 239ms/step - loss: 0.6783 - accuracy: 0.7487 - val_loss: 0.6906 - val_accuracy: 0.7335
Epoch 24/30
169/169 [==============================] - 39s 223ms/step - loss: 0.6465 - accuracy: 0.7572 - val_loss: 0.7422 - val_accuracy: 0.7223
Epoch 25/30
169/169 [==============================] - 39s 223ms/step - loss: 0.6589 - accuracy: 0.7507 - val_loss: 0.7439 - val_accuracy: 0.7261
Epoch 26/30
169/169 [==============================] - 37s 213ms/step - loss: 0.5970 - accuracy: 0.7778 - val_loss: 0.6695 - val_accuracy: 0.7506
Epoch 27/30
169/169 [==============================] - 38s 215ms/step - loss: 0.6279 - accuracy: 0.7695 - val_loss: 0.6938 - val_accuracy: 0.7342
Epoch 28/30
169/169 [==============================] - 38s 222ms/step - loss: 0.5757 - accuracy: 0.7786 - val_loss: 0.6299 - val_accuracy: 0.7743
Epoch 29/30
169/169 [==============================] - 41s 238ms/step - loss: 0.5820 - accuracy: 0.7817 - val_loss: 0.6073 - val_accuracy: 0.7899
```

Visualize the model results

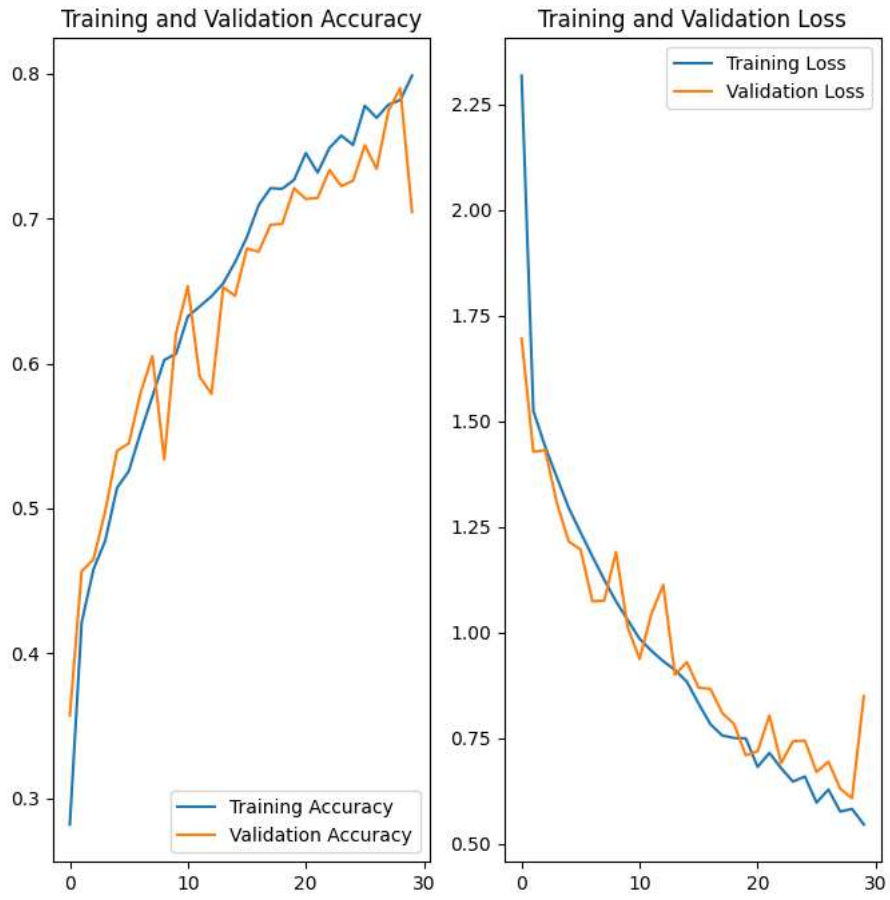
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Analyze your results here. Did you get rid of underfitting/overfitting? Did class rebalance help?

Conclusion:--

We can infer from the above graph that

1. Both training accuracy as well as validation accuracy increases over time as training progresses with each epochs.
2. Both training loss as well as validation loss decreases over time as training progresses with each epochs.
3. Gap/difference between training accuracy and validation accuracy have decreased significantly from previous model, and it has achieved accuracy in the range of **75%-80%** on the validation set.
4. The difference in accuracy between training and validation is very less, around(~1%) which is good. As it indicates that model has learn the hidden facts and has not overfitted.

Class rebalancing not only helps us got rid of overfitting but it also helped in improving the validation accuracy from ~55%(aaprox.) to ~78%(aaprox.).

✓ 0s completed at 6:23 PM