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 that can evaluate images and alert dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

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
import warnings
warnings.filterwarnings('ignore')
```

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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mou
```

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.

```
#/content/gdrive/MyDrive/upgrad/Skin_cancer_ISIC_The_International_Skin_Imaging_Collaboration

!unzip '/content/gdrive/MyDrive/upgrad/Skin_cancer_ISIC_The_International_Skin_Imaging_Collab
```

```
Archive:  /content/gdrive/MyDrive/upgrad/Skin_cancer_ISIC_The_International_Skin_Imag
   creating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
   creating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
   creating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
   creating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
   creating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
  inflating: /content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Internationa
```

```
# Defining the path for train and test images
## Todo: Update the paths of the train and test dataset
data_dir_train = pathlib.Path("/content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The In
data_dir_test = pathlib.Path('/content/gdrive/MyDrive/upgrad/Dataset/Skin cancer ISIC The Int
!ls

     gdrive  sample_data
```

```
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_f
## Note, make sure your resize your images to the size img_height*img_width, while writting t
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_f
## Note, make sure your resize your images to the size img_height*img_width, while writting t
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', 'pi
```

```
print(type(train_ds))
```

```
<class 'tensorflow.python.data.ops.dataset_ops.BatchDataset'>
```

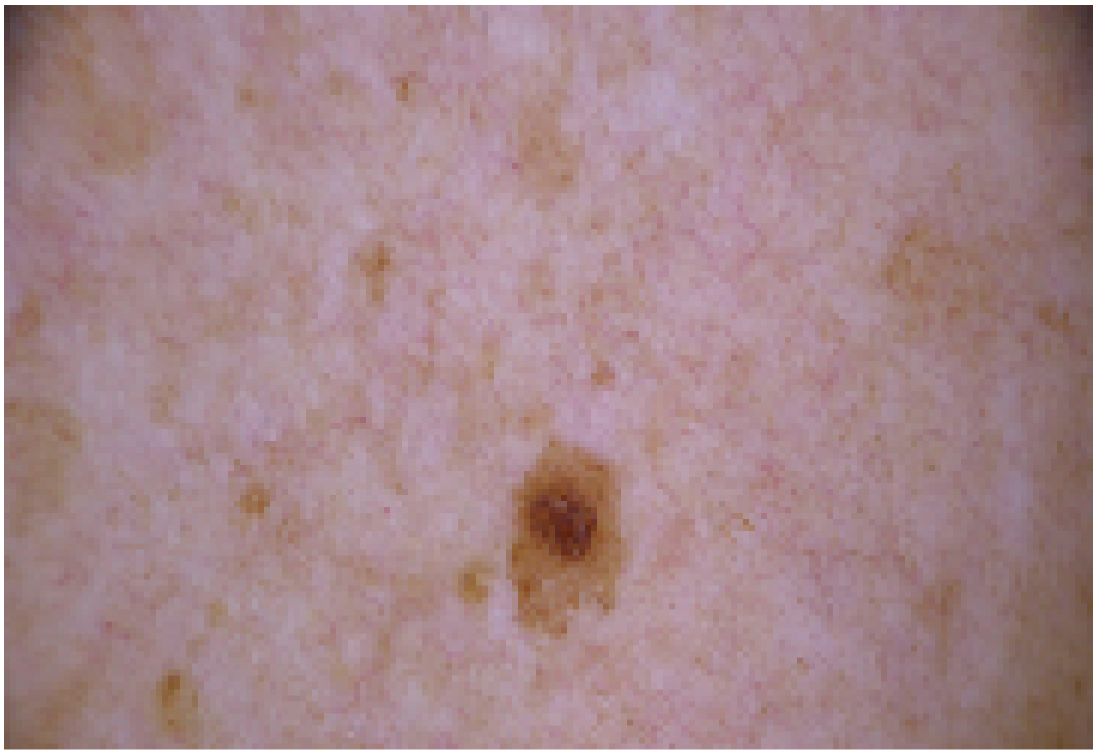
```
for images, labels in train_ds.take(1):
  print(len(images))
  print(len(labels))
```

```
32
32
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  print(len(images))
  print(len(labels))
  plt.imshow(images[0].numpy().astype("uint8"))
  plt.title(class_names[labels[0]])
  plt.axis("off")
```

```
32
32
```

pigmented benign keratosis



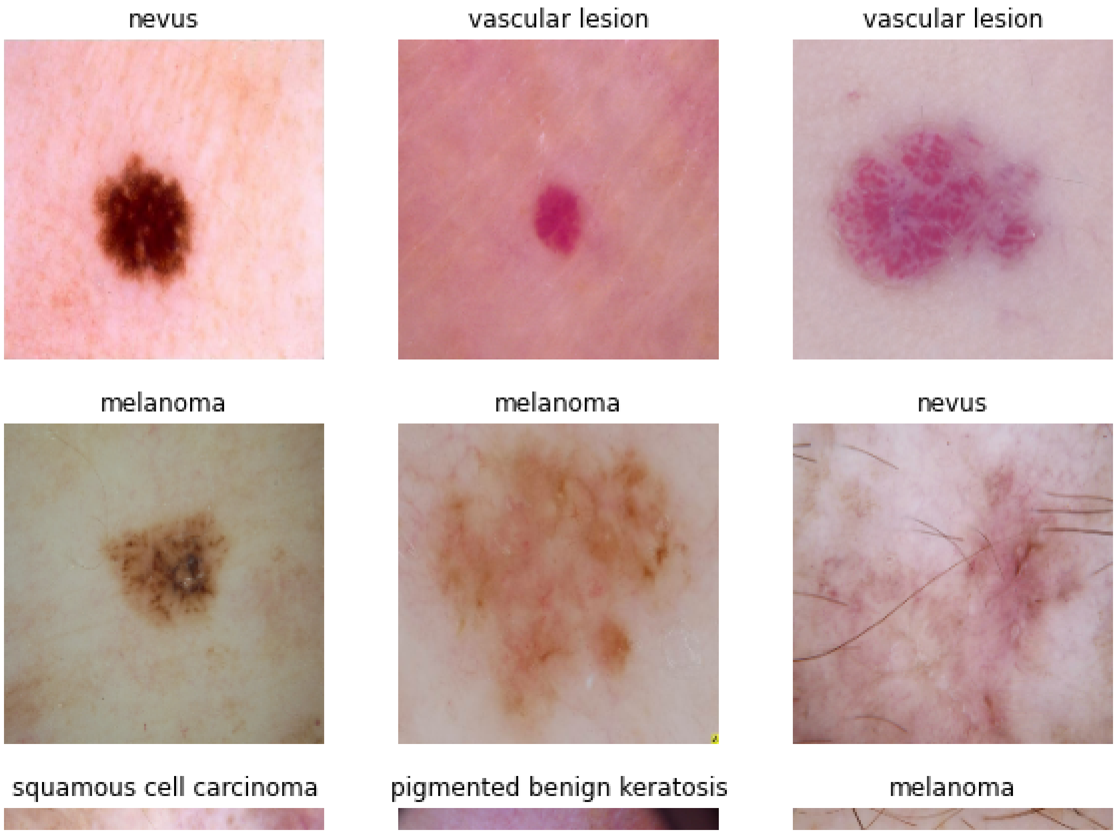
Visualize the data

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

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")

### your code goes here, you can use training or validation data to visualize
```



```
#print(type(train_ds))
#print(len(train_ds)
```

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.

```
#overlaps data preprocessing and model execution while training., Speed up 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 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]`

```
### Your code goes here
num_classes = 9

#A Sequential model is appropriate for a plain stack of layers where each layer has exactly o
model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3))

  #2D convolution layer (e.g. spatial convolution over images).
  layers.Conv2D(16, 3, padding='same', activation='relu'),

  #We slide over the feature map and extract tiles of a specified size.
  #Downsamples the input along its spatial dimensions (height and width) by taking the maximu
  layers.MaxPooling2D(),

  #We slide over the feature map and extract tiles of a specified size.
  layers.Conv2D(32, 3, padding='same', activation='relu'),

  layers.MaxPooling2D(),
  layers.Dropout(0.1),

  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.Dropout(0.1),

  #We slide over the feature map and extract tiles of a specified size.
  #Advantages of downsampling - Decreased size of input for upcoming layers, Works against ov
  layers.MaxPooling2D(),

  #Flattening - Convert into 1D feature vector.  Flattens all its structure to create a singl
  ##Flattens the input. Does not affect the batch size.
  layers.Flatten(),

  #fully connected layer
  #A hidden layer in which each node is connected to every node in the subsequent hidden laye
  #A fully connected layer is also known as a dense layer.

  layers.Dense(128, activation='relu'),

  #Dense is the only actual network layer in that model. A Dense layer feeds all outputs from
  #It's the most basic layer in neural networks. A Dense(10) has ten neurons. A Dense(512) ha
  #Dense implements the operation: output = activation(dot(input, kernel)
  #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
  layers.Dense(num_classes)
  #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
])
```

Compile the model

Choose an appropirate optimiser and loss function for model training

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

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_6 (Conv2D)           (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 90, 90, 16)       0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 90, 90, 32)        4640      
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 45, 45, 32)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 45, 45, 32)        0         
                                                                 
 conv2d_8 (Conv2D)           (None, 45, 45, 64)        18496     
                                                                 
 dropout_2 (Dropout)         (None, 45, 45, 64)        0         
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 22, 22, 64)       0         
 2D)                                                             
                                                                 
 flatten_2 (Flatten)         (None, 30976)             0         
                                                                 
 dense_4 (Dense)             (None, 128)               3965056   
                                                                 
 dense_5 (Dense)             (None, 9)                 1161      
                                                                 
=================================================================
Total params: 3,989,801
Trainable params: 3,989,801
Non-trainable params: 0
_________________________________________________________________
```

```
### Todo, choose an appropirate optimiser and loss function
#RMSprop. RMSprop is a very effective, but currently unpublished adaptive learning rate metho
#Adam. Adam is a recently proposed update that looks a bit like RMSProp with momentum. The (s
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

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

```
Epoch 1/20
56/56 [==============================] - 4s 36ms/step - loss: 2.0564 - accuracy: 0.2785
Epoch 2/20
56/56 [==============================] - 2s 31ms/step - loss: 1.6457 - accuracy: 0.4235
Epoch 3/20
56/56 [==============================] - 2s 31ms/step - loss: 1.5058 - accuracy: 0.4704
Epoch 4/20
56/56 [==============================] - 2s 30ms/step - loss: 1.4128 - accuracy: 0.5050
Epoch 5/20
56/56 [==============================] - 2s 29ms/step - loss: 1.3185 - accuracy: 0.5340
Epoch 6/20
56/56 [==============================] - 2s 31ms/step - loss: 1.2714 - accuracy: 0.5502
Epoch 7/20
56/56 [==============================] - 2s 31ms/step - loss: 1.1749 - accuracy: 0.5848
Epoch 8/20
56/56 [==============================] - 2s 30ms/step - loss: 1.1156 - accuracy: 0.6032
Epoch 9/20
56/56 [==============================] - 2s 30ms/step - loss: 1.0398 - accuracy: 0.6166
Epoch 10/20
56/56 [==============================] - 2s 29ms/step - loss: 0.9498 - accuracy: 0.6535
Epoch 11/20
56/56 [==============================] - 2s 30ms/step - loss: 0.9256 - accuracy: 0.6629
Epoch 12/20
56/56 [==============================] - 2s 30ms/step - loss: 0.8247 - accuracy: 0.7009
Epoch 13/20
56/56 [==============================] - 2s 42ms/step - loss: 0.7530 - accuracy: 0.7210
Epoch 14/20
56/56 [==============================] - 2s 37ms/step - loss: 0.7725 - accuracy: 0.7199
Epoch 15/20
56/56 [==============================] - 2s 30ms/step - loss: 0.6511 - accuracy: 0.7561
Epoch 16/20
56/56 [==============================] - 2s 30ms/step - loss: 0.6135 - accuracy: 0.7773
Epoch 17/20
56/56 [==============================] - 2s 30ms/step - loss: 0.5524 - accuracy: 0.7991
Epoch 18/20
56/56 [==============================] - 2s 30ms/step - loss: 0.5176 - accuracy: 0.8097
Epoch 19/20
56/56 [==============================] - 2s 30ms/step - loss: 0.4997 - accuracy: 0.8192
Epoch 20/20
56/56 [==============================] - 2s 32ms/step - loss: 0.4097 - accuracy: 0.8482
```

Train the model

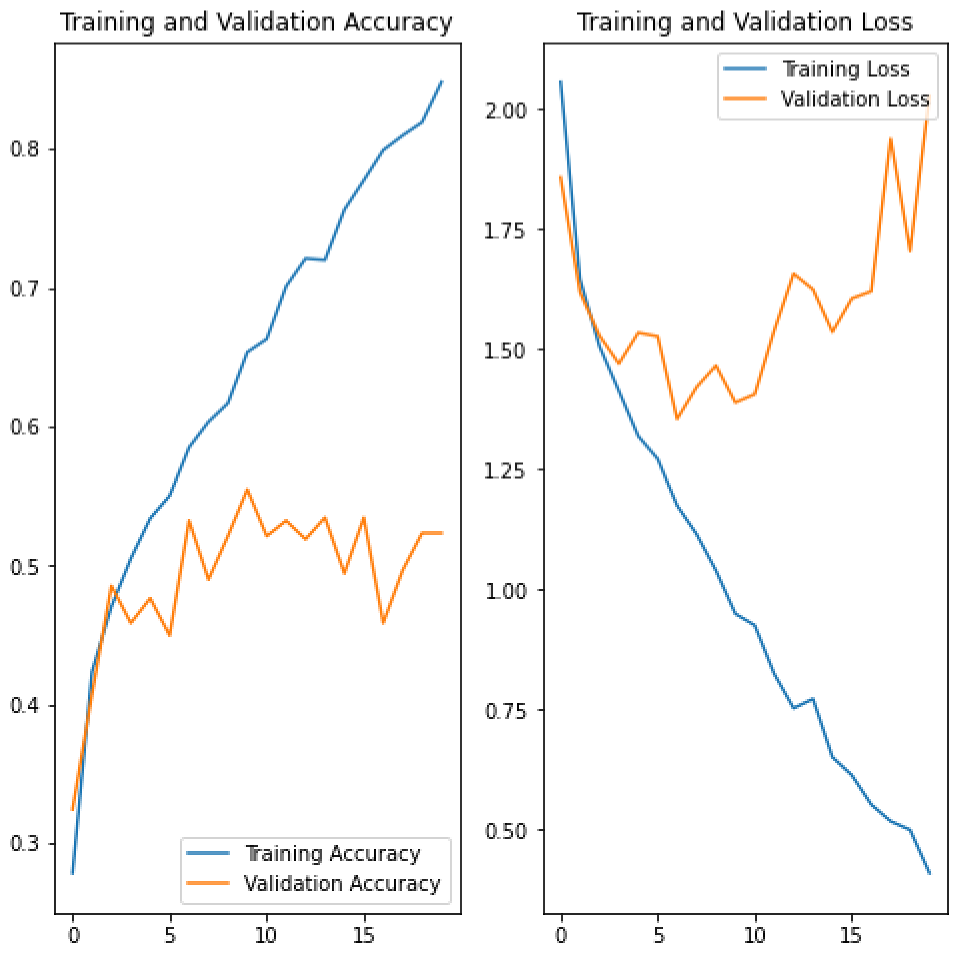
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



```
### Your code goes here
num_classes = 9
```

```
model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3))
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  #We slide over the feature map and extract tiles of a specified size.
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  #We slide over the feature map and extract tiles of a specified size.
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  #We slide over the feature map and extract tiles of a specified size.
  layers.MaxPooling2D(),
  #Advantages of downsampling - Decreased size of input for upcoming layers, Works against ov
  layers.Flatten(),
  #Flattening - Convert into 1D feature vector.  Flattens all its structure to create a singl
  layers.Dense(128, activation='relu'),
  #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
  layers.Dense(num_classes)
  #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
])
```

Visualizing training results

Todo: Write your findings after the model fit, see if there is an evidence of model overfit or underfit

Write your findings here

```
# Todo, after you have analysed the model fit history for presence of underfit or overfit, ch
data_augmentation = keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal",
                                                 input_shape=(img_height,
                                                              img_width,
                                                              3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
    layers.experimental.preprocessing.RandomTranslation(1,.5,fill_mode="reflect",interpolatio
    layers.experimental.preprocessing.RandomCrop(img_height,img_width),
  ]
)
```

```
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
```

```
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause th
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause th
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 cause t
```

```
# Todo, visualize how your augmentation strategy works for one instance of training image.
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 caus
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
WARNING:tensorflow:5 out of the last 5 calls to <function pfor.<locals>.f at 0x7fcde5
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
WARNING:tensorflow:6 out of the last 6 calls to <function pfor.<locals>.f at 0x7fcde5
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 caus
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
```

Todo:

Create the model, compile and train the model

```
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
```

```
## You can use Dropout layer if there is an evidence of overfitting in your findings

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(256, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause th
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause th
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause ther
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no re
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registere
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 cause t
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
```

Compiling the model

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
```

```
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
```

```
## Your code goes here
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
```

Training the model

```
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
```

```
## Your code goes here, note: train your model for 20 epochs
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
    Epoch 1/20
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 caus
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
    WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause
    WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
    WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
```

```
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is nc
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause t
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is nc
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformIntV2 caus
56/56 [==============================] - 38s 514ms/step - loss: 2.1566 - accuracy: 0.
Epoch 2/20
56/56 [==============================] - 28s 494ms/step - loss: 1.9958 - accuracy: 0.
Epoch 3/20
56/56 [==============================] - 27s 484ms/step - loss: 1.8390 - accuracy: 0.
Epoch 4/20
56/56 [==============================] - 27s 487ms/step - loss: 1.7010 - accuracy: 0.
Epoch 5/20
56/56 [==============================] - 27s 489ms/step - loss: 1.6554 - accuracy: 0.
Epoch 6/20
```

Visualizing the results

```
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no regist
```

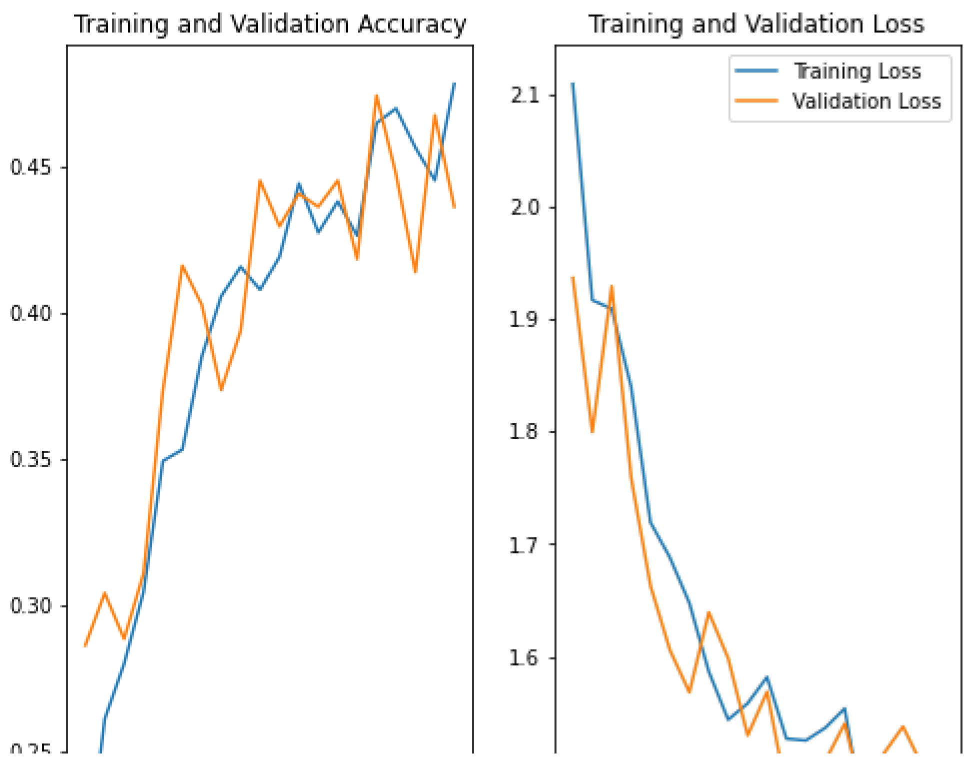
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



```
# For convenience, let us set up the path for the training and validation sets
train_dir = os.path.join('/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Ima
val_dir = os.path.join('/content/gdrive/MyDrive/Skin cancer ISIC The International Skin Imagi
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Setting batch size and image size
batch_size = 100
IMG_SHAPE = 224

# Create training images generator
#Generate batches of tensor image data with real-time data augmentation.
#https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )
#https://keras.io/api/preprocessing/image/
#Then calling image_dataset_from_directory(main_directory, labels='inferred') will return a t
train_data_gen = image_gen_train.flow_from_directory(
                                                    batch_size=batch_size,
                                                    directory=train_dir,
                                                    shuffle=True,
                                                    target_size=(IMG_SHAPE,IMG_SHAPE),
                                                    class_mode='sparse'
                                                    )
```

```
# Create validation images generator
image_gen_val = ImageDataGenerator(rescale=1./255)
val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                 directory=val_dir,
                                                 target_size=(IMG_SHAPE, IMG_SHAPE),
                                                 class_mode='sparse')
```

```
     Found 2239 images belonging to 9 classes.
     Found 118 images belonging to 9 classes.
```

```
#Create a CNN model
#Experiment #1
#A Sequential model is appropriate for a plain stack of layers where each layer has exactly o
import numpy as np
import glob
import shutil
import matplotlib.pyplot as plt

# Import layers explicitly to keep our code compact
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D


model = Sequential()

#2D convolution layer (e.g. spatial convolution over images).
model.add(Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_SHAPE,IMG_SHAPE,
#Downsamples the input along its spatial dimensions (height and width) by taking the maximum
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, 3, padding='same', activation='relu'))

#Downsamples the input along its spatial dimensions (height and width) by taking the maximum
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, 3, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

#Flattens the input. Does not affect the batch size.
model.add(Flatten())

#https://keras.io/api/layers/regularization_layers/dropout/
#The Dropout layer randomly sets input units to 0 with a frequency of rate at each step durin
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))

model.add(Dropout(0.2))

#Just your regular densely-connected NN layer.
```

```
#Dense is the only actual network layer in that model. A Dense layer feeds all outputs from t
#It's the most basic layer in neural networks. A Dense(10) has ten neurons. A Dense(512) has
#Dense implements the operation: output = activation(dot(input, kernel)
model.add(Dense(9))

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
epochs = 20

history = model.fit(
  train_data_gen,
  validation_data=val_data_gen,
  epochs=10
)
```

```
Epoch 1/10
23/23 [==============================] - 121s 5s/step - loss: 3.3523 - accuracy: 0.1831
Epoch 2/10
23/23 [==============================] - 76s 3s/step - loss: 1.8302 - accuracy: 0.3247 -
Epoch 3/10
23/23 [==============================] - 76s 3s/step - loss: 1.6664 - accuracy: 0.3872 -
Epoch 4/10
23/23 [==============================] - 76s 3s/step - loss: 1.6388 - accuracy: 0.4006 -
Epoch 5/10
23/23 [==============================] - 78s 3s/step - loss: 1.5575 - accuracy: 0.4466 -
Epoch 6/10
23/23 [==============================] - 76s 3s/step - loss: 1.4707 - accuracy: 0.4766 -
Epoch 7/10
23/23 [==============================] - 75s 3s/step - loss: 1.4547 - accuracy: 0.4944 -
Epoch 8/10
23/23 [==============================] - 77s 3s/step - loss: 1.3962 - accuracy: 0.5190 -
Epoch 9/10
23/23 [==============================] - 76s 3s/step - loss: 1.3968 - accuracy: 0.5051 -
Epoch 10/10
23/23 [==============================] - 76s 3s/step - loss: 1.3417 - accuracy: 0.5315 -
```

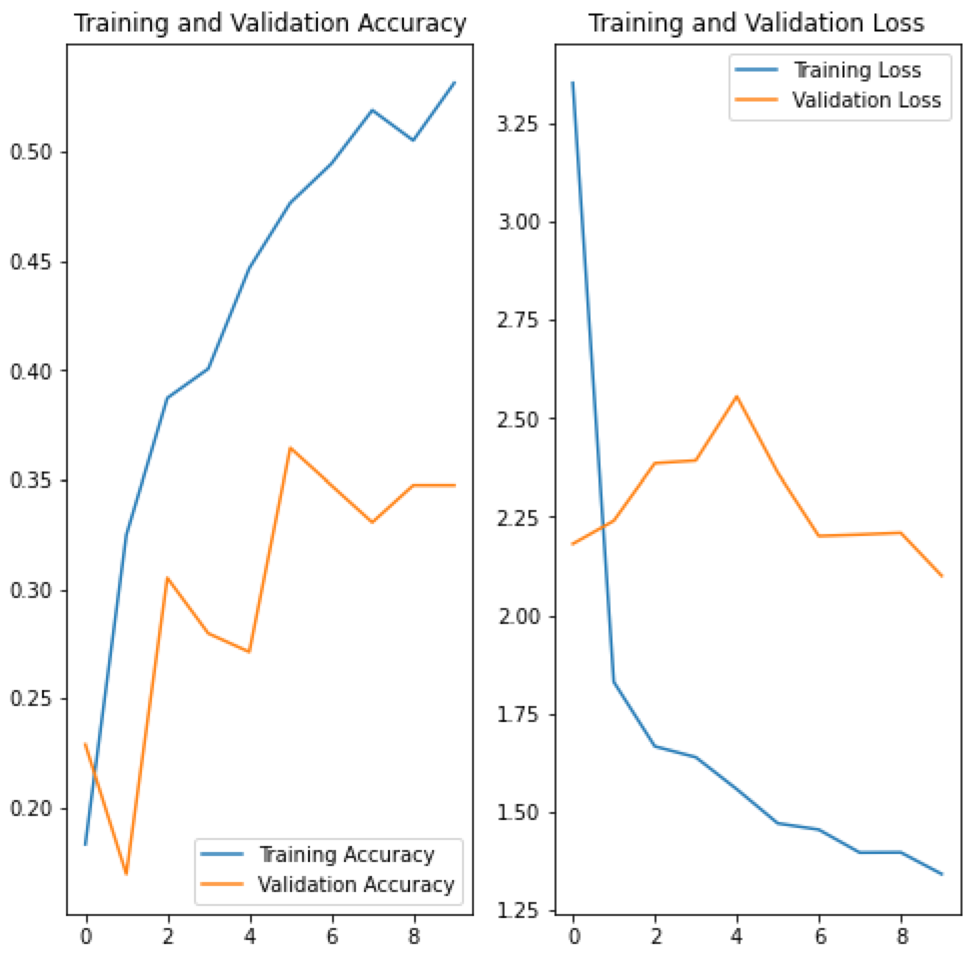
```
import matplotlib.pyplot as plt
epochs=10
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Todo: Write your findings after the model fit, see if there is an evidence of model overfit or underfit. Do you think there is some improvement now as compared to the previous model run?

Todo: 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.

```
## Your code goes here.
from glob import glob
path_list = [x for x in glob(os.path.join(data_dir_train, '*', '*.jpg'))]
lesion_list = [os.path.basename(os.path.dirname(y)) for y in glob(os.path.join(data_dir_train
len(path_list)
```

```
    2239
```

```
dataframe_dict_original = dict(zip(path_list, lesion_list))
original_df = pd.DataFrame(list(dataframe_dict_original.items()),columns = ['Path','Label'])
original_df
```

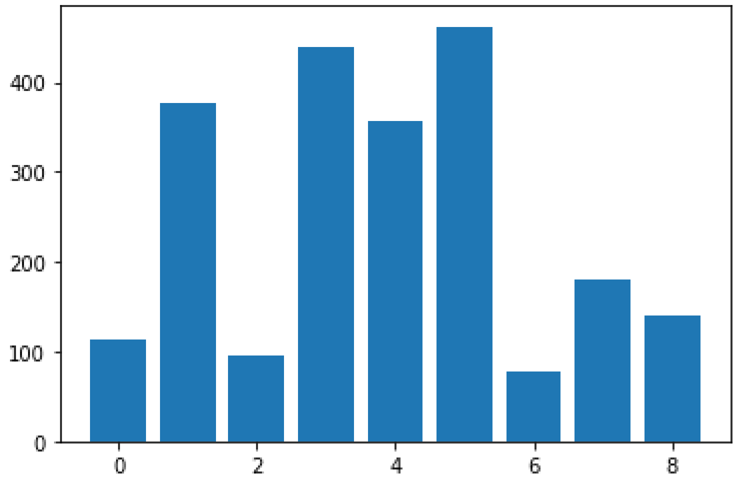
	Path	Label
0	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	actinic keratosis
1	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	actinic keratosis
2	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	actinic keratosis
3	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	actinic keratosis
4	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	actinic keratosis
...	...	...
2234	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	vascular lesion
2235	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	vascular lesion
2236	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	vascular lesion
2237	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	vascular lesion
2238	/content/gdrive/MyDrive/upgrad/Dataset/Skin ca...	vascular lesion

2239 rows × 2 columns

```
from sklearn.preprocessing import LabelEncoder
from collections import Counter
# split into input and output elements
X, y = original_df['Path'], original_df['Label']
# label encode the target variable
y = LabelEncoder().fit_transform(y)
# summarize distribution
counter = Counter(y)
```

```
for k,v in counter.items():
    per = v / len(y) * 100
    print('Class=%d, n=%d (%.3f%%)' % (k, v, per))
# plot the distribution
plt.bar(counter.keys(), counter.values())
plt.show()
```

```
Class=0, n=114 (5.092%)
Class=1, n=376 (16.793%)
Class=2, n=95 (4.243%)
Class=3, n=438 (19.562%)
Class=4, n=357 (15.945%)
Class=5, n=462 (20.634%)
Class=6, n=77 (3.439%)
Class=7, n=181 (8.084%)
Class=8, n=139 (6.208%)
```



Todo: Write your findings here:

- Which class has the least number of samples?

- Which classes dominate the data in terms proportionate number of samples?

Todo: 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.

```
#https://datascience.stackexchange.com/questions/13490/how-to-set-class-weights-for-imbalance

from sklearn.utils import class_weight
#Class=0, n=114 (5.092%)
#Class=1, n=376 (16.793%)
#Class=2, n=95 (4.243%)
```

```
#Class=3, n=438 (19.562%)
#Class=4, n=357 (15.945%)
#Class=5, n=462 (20.634%)
#Class=6, n=77 (3.439%)
#Class=7, n=181 (8.084%)
#Class=8, n=139 (6.208%)

class_weight = {0:5.09,
                1:16.79,
                2:4.24,
                3:19.56,
                4:15.94,
                5:20.63,
                6:3.43,
                7:8.08,
                8:6.20}

#class_weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y_train)

### Your code goes here
num_classes = 9

#A Sequential model is appropriate for a plain stack of layers where each layer has exactly o
model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3))

  #2D convolution layer (e.g. spatial convolution over images).
  layers.Conv2D(16, 3, padding='same', activation='relu'),

  #We slide over the feature map and extract tiles of a specified size.
  #Downsamples the input along its spatial dimensions (height and width) by taking the maximu
  layers.MaxPooling2D(),

  #We slide over the feature map and extract tiles of a specified size.
  layers.Conv2D(32, 3, padding='same', activation='relu'),

  layers.MaxPooling2D(),
  layers.Dropout(0.1),

  layers.Conv2D(64, 3, padding='same', activation='relu'),

  #We slide over the feature map and extract tiles of a specified size.
  #Advantages of downsampling - Decreased size of input for upcoming layers, Works against ov
  layers.MaxPooling2D(),
  layers.Dropout(0.1),

  #Flattening - Convert into 1D feature vector.  Flattens all its structure to create a singl
  ##Flattens the input. Does not affect the batch size.
  layers.Flatten(),

  #fully connected layer
```

```
    #A hidden layer in which each node is connected to every node in the subsequent hidden laye
    #A fully connected layer is also known as a dense layer.

    layers.Dense(128, activation='relu'),

    #Dense is the only actual network layer in that model. A Dense layer feeds all outputs from
    #It's the most basic layer in neural networks. A Dense(10) has ten neurons. A Dense(512) ha
    #Dense implements the operation: output = activation(dot(input, kernel)
    #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
    layers.Dense(num_classes)
    #Dense Layer - A dense layer represents a matrix vector multiplication.  each input node is
  ])


  ### Todo, choose an appropirate optimiser and loss function
  #RMSprop. RMSprop is a very effective, but currently unpublished adaptive learning rate metho
  #Adam. Adam is a recently proposed update that looks a bit like RMSProp with momentum. The (s
  model.compile(optimizer='adam',
                loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                metrics=['accuracy'])

  epochs = 20
  history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    class_weight=class_weight)


      Epoch 1/20
      56/56 [==============================] - 2s 27ms/step - loss: 25.9302 - accuracy: 0.264
      Epoch 2/20
      56/56 [==============================] - 1s 24ms/step - loss: 21.5376 - accuracy: 0.370
      Epoch 3/20
      56/56 [==============================] - 1s 23ms/step - loss: 20.0326 - accuracy: 0.411
      Epoch 4/20
      56/56 [==============================] - 1s 24ms/step - loss: 17.7326 - accuracy: 0.500
      Epoch 5/20
      56/56 [==============================] - 1s 24ms/step - loss: 17.4160 - accuracy: 0.491
      Epoch 6/20
      56/56 [==============================] - 1s 24ms/step - loss: 17.6267 - accuracy: 0.492
      Epoch 7/20
      56/56 [==============================] - 1s 23ms/step - loss: 15.6782 - accuracy: 0.535
      Epoch 8/20
      56/56 [==============================] - 1s 24ms/step - loss: 14.5885 - accuracy: 0.579
      Epoch 9/20
      56/56 [==============================] - 1s 24ms/step - loss: 13.9129 - accuracy: 0.583
      Epoch 10/20
      56/56 [==============================] - 2s 29ms/step - loss: 13.0691 - accuracy: 0.594
      Epoch 11/20
      56/56 [==============================] - 1s 25ms/step - loss: 11.3796 - accuracy: 0.635
      Epoch 12/20
      56/56 [==============================] - 1s 24ms/step - loss: 10.8046 - accuracy: 0.652
      Epoch 13/20
```

```
56/56 [==============================] - 1s 24ms/step - loss: 10.2840 - accuracy: 0.666
Epoch 14/20
56/56 [==============================] - 1s 24ms/step - loss: 9.1983 - accuracy: 0.7037
Epoch 15/20
56/56 [==============================] - 1s 25ms/step - loss: 7.8667 - accuracy: 0.7277
Epoch 16/20
56/56 [==============================] - 1s 24ms/step - loss: 8.0918 - accuracy: 0.7372
Epoch 17/20
56/56 [==============================] - 1s 24ms/step - loss: 6.1371 - accuracy: 0.7835
Epoch 18/20
56/56 [==============================] - 1s 24ms/step - loss: 5.4177 - accuracy: 0.8047
Epoch 19/20
56/56 [==============================] - 1s 24ms/step - loss: 4.3347 - accuracy: 0.8421
Epoch 20/20
56/56 [==============================] - 1s 24ms/step - loss: 4.3947 - accuracy: 0.8387
```

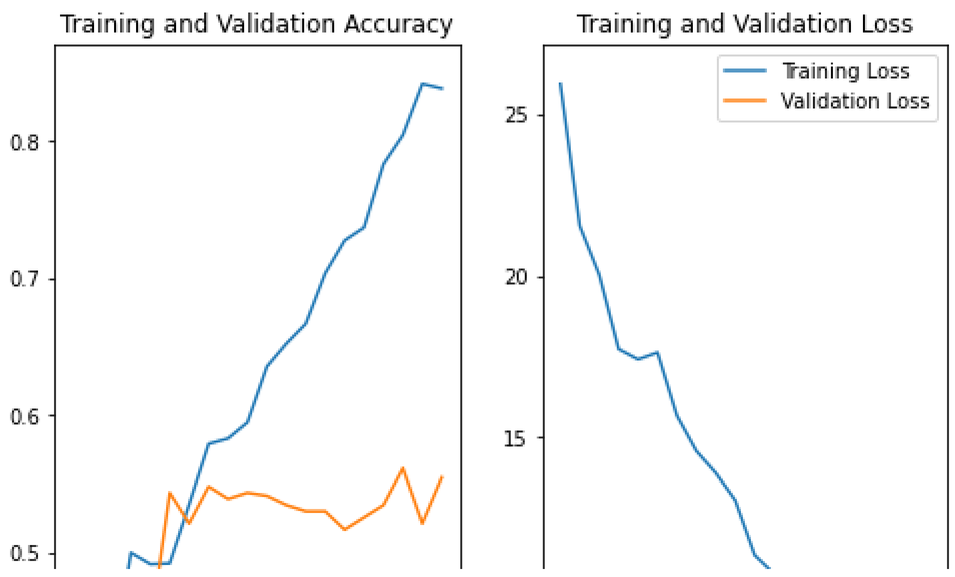
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



```
!pip install Augmentor
```

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

```
#https://github.com/mdbloice/Augmentor
#https://github.com/mdbloice/Augmentor
datapath = '/content/gdrive/MyDrive/upgrad/Dataset/SKC/Train/seborrheic keratosis'
import Augmentor
p = Augmentor.Pipeline(datapath)
#Every function requires you to specify a probability, which is used to decide if an operatio
p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
#p.zoom(probability=0.5, min_factor=1.1, max_factor=1.5)
p.sample(200)
p.process()
```

```
Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/upgrad/Dataset/SKC/Train/seborrheic kera
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1024x768 at 0x7FCDE53A
```

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="/content/gdrive/MyDrive/SC/Train//"
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 classe
```

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)
```

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

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

```
['Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0031217.jpg_7f885971-40de-416f-93f8-
4d940c80ba90.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0025321.jpg_c380f0d6-4c6c-4974-93d7-
cf7ca3c69b3f.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0028431.jpg_bcc4e007-ce77-44c2-9e99-
0607dba4e661.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0027269.jpg_b6fd7adb-e500-414d-a638-
4ead69c06196.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0030606.jpg_c888632b-43d9-4bb1-844b-
5f5347489894.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0032557.jpg_304e812f-0f26-4206-91bd-
f36142d97356.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0033092.jpg_a5c96a3a-7a90-4b9d-ae24-
82b0a5a14345.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0030070.jpg_50f40a64-0933-454a-bc3c-
f361cbdac1b9.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0033135.jpg_230e837f-7592-497e-94be-
3c5c2f047b5c.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0033230.jpg_038c6771-b4b9-452b-9a24-
19285e10ba81.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0028188.jpg_56f4ff48-f856-4fd5-92c0-
33ef7de8afa3.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0028714.jpg_d3d2cebf-bc74-45fa-857b-
b063729803f3.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0027522.jpg_a2d5e91d-f18e-4142-b848-
47da49001b26.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0031090.jpg_f4e98991-ef03-4be7-9b70-
5a1d6d4cd5b9.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0026349.jpg_2e2f747b-4150-4d7f-8b05-
c434710dcf8b.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0029877.jpg_cf68d697-c861-41e0-aeca-
972fe84ee8ae.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0028885.jpg_040eb75b-22da-4e7d-8e8e-
418c92144478.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
lesion/output/vascular lesion_original_ISIC_0032409.jpg_4f19d65a-cc80-4e7b-8f64-
c9a0fcc5ba36.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular
```

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

```
['vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
 'vascular lesion',
```

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

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

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

```
    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
```

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.

Todo: Train the model on the data created using Augmentor

[ ] ↳ *1 cell hidden*

Todo: Create a training dataset

[ ] ↳ *1 cell hidden*

Todo: Create a validation dataset

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  validation_split = 0.2,
  subset = 'validation',## Todo choose the correct parameter value, so that only validation d
  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)

```
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)

model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

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

```
## your code goes here
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Todo: Train your model

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

```
Epoch 1/20
169/169 [==============================] - 11s 62ms/step - loss: 1.9140 - accuracy: 0.2
Epoch 2/20
169/169 [==============================] - 5s 28ms/step - loss: 1.4312 - accuracy: 0.45
Epoch 3/20
169/169 [==============================] - 5s 28ms/step - loss: 1.2410 - accuracy: 0.53
Epoch 4/20
169/169 [==============================] - 5s 28ms/step - loss: 1.0454 - accuracy: 0.62
Epoch 5/20
169/169 [==============================] - 5s 28ms/step - loss: 0.8814 - accuracy: 0.68
Epoch 6/20
169/169 [==============================] - 5s 28ms/step - loss: 0.7177 - accuracy: 0.73
Epoch 7/20
169/169 [==============================] - 5s 28ms/step - loss: 0.5872 - accuracy: 0.78
Epoch 8/20
169/169 [==============================] - 5s 28ms/step - loss: 0.4969 - accuracy: 0.82
Epoch 9/20
169/169 [==============================] - 5s 28ms/step - loss: 0.4206 - accuracy: 0.85
Epoch 10/20
169/169 [==============================] - 5s 28ms/step - loss: 0.3517 - accuracy: 0.87
Epoch 11/20
169/169 [==============================] - 5s 28ms/step - loss: 0.3264 - accuracy: 0.87
Epoch 12/20
169/169 [==============================] - 5s 28ms/step - loss: 0.2863 - accuracy: 0.89
Epoch 13/20
169/169 [==============================] - 5s 28ms/step - loss: 0.2640 - accuracy: 0.90
Epoch 14/20
169/169 [==============================] - 5s 28ms/step - loss: 0.2817 - accuracy: 0.89
Epoch 15/20
169/169 [==============================] - 5s 28ms/step - loss: 0.2236 - accuracy: 0.91
Epoch 16/20
169/169 [==============================] - 5s 29ms/step - loss: 0.1791 - accuracy: 0.93
```

```
Epoch 17/20
169/169 [==============================] - 5s 28ms/step - loss: 0.1865 - accuracy: 0.92
Epoch 18/20
169/169 [==============================] - 5s 28ms/step - loss: 0.1616 - accuracy: 0.94
Epoch 19/20
169/169 [==============================] - 5s 28ms/step - loss: 0.1845 - accuracy: 0.92
Epoch 20/20
169/169 [==============================] - 5s 28ms/step - loss: 0.1387 - accuracy: 0.94
```

Todo: Visualize the model results

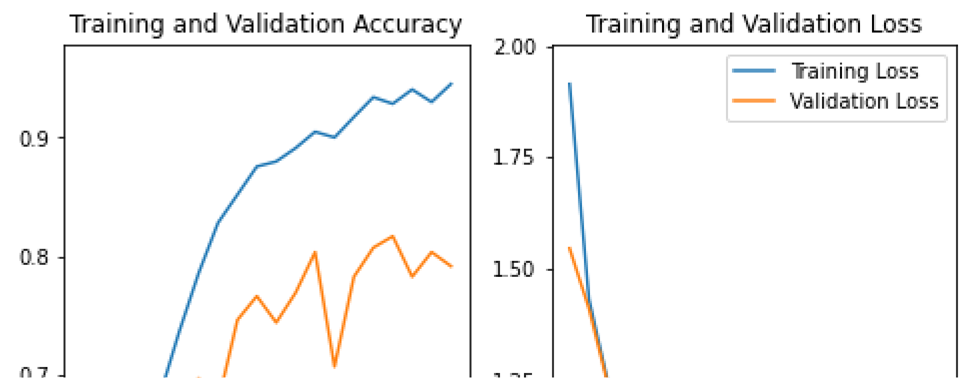
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



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

The class rebalance helped in reducing overfititng of the data and thus the loass is beng reduced But it reduced the Acurracy very low

Initially we tried without the ImageDataGenerator which created data to over fit at high ratio

Then we introduced dropout and ImageDataGenerator which reduced the over fit