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.

The dataset consists of 2357 images of malignant and benign oncological diseases, which were formed from the International Skin Imaging Collaboration (ISIC). All images were sorted according to the classification taken with ISIC, and all subsets were divided into the same number of images, with the exception of melanomas and moles, whose images are slightly dominant.

The data set contains the following diseases:

Actinic keratosis

Basal cell carcinoma

Dermatofibroma

Melanoma

Nevus

Pigmented benign keratosis

Seborrheic keratosis

Squamous cell carcinoma

Vascular lesion

```
# Import all the important libraries


import pathlib
import tensorflow as tf
import matplotlib as mat
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

from __future__ import print_function
from tensorflow.keras.layers import BatchNormalization
from tensorflow.python.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.utils.np_utils import to_categorical
from tensorflow.keras.regularizers import l2


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

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

Data Reading/Data Understanding:

```
# Defining the path for train and test images

data_dir_train = pathlib.Path("/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaborat
data_dir_test = pathlib.Path("/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaborati
```

```
# distribution of test and train images
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
```

Dataset Creation

```
#batch size
batch_size = 32
img_height = 180
img_width = 180
```

Create train & validation dataset from the train directory

```
## 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,
  labels='inferred',
  validation_split=0.2,
  color_mode='rgb',
  subset="training",
  seed=123,
  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,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

```
#test dataset
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_test,
  labels='inferred',
  color_mode='rgb',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 118 files belonging to 9 classes.
```

```
# 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
```

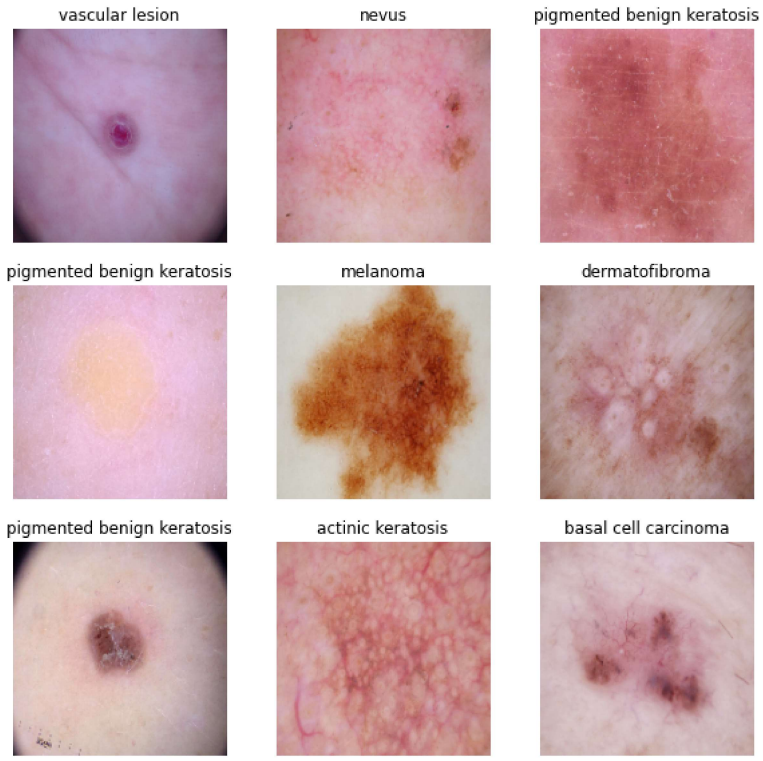
Dataset visualisation

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")
```



The image_batch is a tensor of the shape (32, 180, 180, 3). This is a batch of 32 images of shape 180x180x3 (3 referes to channel 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 Model

rescale images to normalize pixel values between (0,1).

```
preprocessing_layers = [
    tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(180, 180, 3))
]
```

*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]

CNN Model

```
input_shape = (180,180,3)
lr = 1e-5
init = 'normal'
```

```
activ = 'relu'

model = Sequential()
#normalisation layer
model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(180, 180, 3)))
# Adding Conv layers
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding ='same' ,input_shape=input_shape))
model.add(MaxPool2D(pool_size=(2, 2)))

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

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

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding ='same'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
#model.add(Dropout(0.5))
model.add(Dense(9,kernel_regularizer=l2(0.01)))
model.add(Activation('softmax'))

## Number of classes is 9
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

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

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

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

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

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

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

 module_wrapper_6 (ModuleWra  (None, 22, 22, 256)      295168
 pper)

 module_wrapper_7 (ModuleWra  (None, 11, 11, 256)      0
 pper)

 module_wrapper_8 (ModuleWra  (None, 30976)            0
 pper)

 module_wrapper_9 (ModuleWra  (None, 512)              15860224
 pper)

 module_wrapper_10 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_11 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 16,253,257
Trainable params: 16,253,257
Non-trainable params: 0
_________________________________________________________________
```

Compile the model

```
### Todo, choose an appropirate optimiser and loss function
from tensorflow.keras.optimizers import Adam
```

```
optimizer_1 = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
loss_function_1 = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer_1,
              loss=loss_function_1,
              metrics=['accuracy'])



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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

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

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

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

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

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

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

 module_wrapper_6 (ModuleWra  (None, 22, 22, 256)      295168
 pper)

 module_wrapper_7 (ModuleWra  (None, 11, 11, 256)      0
 pper)

 module_wrapper_8 (ModuleWra  (None, 30976)            0
 pper)

 module_wrapper_9 (ModuleWra  (None, 512)              15860224
 pper)

 module_wrapper_10 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_11 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 16,253,257
Trainable params: 16,253,257
Non-trainable params: 0
_________________________________________________________________
```

```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
    patience=3,
    verbose=1,
    factor=0.5,
    min_lr=0.00001)



epochs = 1
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,callbacks=[learning_rate_reduction]
)
```

```
/usr/local/lib/python3.8/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received
  return dispatch_target(*args, **kwargs)
56/56 [==============================] - 216s 860ms/step - loss: 1.9817 - accuracy: 0.2528 - val_loss: 2.0095 - val_accuracy: 0.2394 - l
```

The model accuracies are:

train -- 25.28%

val -- 23.94

Which is a very poor model with a high underfitting

Create model with dropout to address the overfitting

Dropout and L2 regularization are used as augmentation to prevent the overfitting

```
input_shape = (180,180,3)
lr = 1e-5
init = 'normal'
activ = 'relu'

model = Sequential()
#normalisation layer
model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(180, 180, 3)))
# Adding Conv layers
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding ='same' ,input_shape=input_shape))
model.add(MaxPool2D(pool_size=(2, 2)))

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

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

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding ='same'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(9,kernel_regularizer=l2(0.01)))
model.add(Activation('softmax'))

## Number of classes is 9
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 module_wrapper_12 (ModuleWr  (None, 180, 180, 32)     896
 apper)

 module_wrapper_13 (ModuleWr  (None, 90, 90, 32)       0
 apper)

 module_wrapper_14 (ModuleWr  (None, 90, 90, 64)       18496
 apper)

 module_wrapper_15 (ModuleWr  (None, 45, 45, 64)       0
 apper)

 module_wrapper_16 (ModuleWr  (None, 45, 45, 128)      73856
 apper)

 module_wrapper_17 (ModuleWr  (None, 22, 22, 128)      0
 apper)

 module_wrapper_18 (ModuleWr  (None, 22, 22, 256)      295168
 apper)

 module_wrapper_19 (ModuleWr  (None, 11, 11, 256)      0
 apper)

 module_wrapper_20 (ModuleWr  (None, 30976)            0
 apper)

 module_wrapper_21 (ModuleWr  (None, 512)              15860224
 apper)

 module_wrapper_22 (ModuleWr  (None, 512)              0
```

```
 apper)

 module_wrapper_23 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_24 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 16,253,257
Trainable params: 16,253,257
Non-trainable params: 0
_________________________________________________________________
```

```
### Todo, choose an appropirate optimiser and loss function
from tensorflow.keras.optimizers import Adam
optimizer_1 = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
loss_function_1 = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer_1,
              loss=loss_function_1,
              metrics=['accuracy'])
```

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 module_wrapper_12 (ModuleWr  (None, 180, 180, 32)     896
 apper)

 module_wrapper_13 (ModuleWr  (None, 90, 90, 32)       0
 apper)

 module_wrapper_14 (ModuleWr  (None, 90, 90, 64)       18496
 apper)

 module_wrapper_15 (ModuleWr  (None, 45, 45, 64)       0
 apper)

 module_wrapper_16 (ModuleWr  (None, 45, 45, 128)      73856
 apper)

 module_wrapper_17 (ModuleWr  (None, 22, 22, 128)      0
 apper)

 module_wrapper_18 (ModuleWr  (None, 22, 22, 256)      295168
 apper)

 module_wrapper_19 (ModuleWr  (None, 11, 11, 256)      0
 apper)

 module_wrapper_20 (ModuleWr  (None, 30976)            0
 apper)

 module_wrapper_21 (ModuleWr  (None, 512)              15860224
 apper)

 module_wrapper_22 (ModuleWr  (None, 512)              0
 apper)

 module_wrapper_23 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_24 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 16,253,257
Trainable params: 16,253,257
Non-trainable params: 0
_________________________________________________________________
```

```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
```

```
    patience=3,
    verbose=1,
    factor=0.5,
    min_lr=0.00001)
```

Train the model for ~20 epoch

```
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,callbacks=[learning_rate_reduction]
)
```

```
Epoch 1/20
56/56 [==============================] - 5s 65ms/step - loss: 2.0241 - accuracy: 0.2260 - val_loss: 1.8685 - val_accuracy: 0.3289 - lr:
Epoch 2/20
56/56 [==============================] - 3s 47ms/step - loss: 1.8360 - accuracy: 0.3136 - val_loss: 1.7755 - val_accuracy: 0.3535 - lr:
Epoch 3/20
56/56 [==============================] - 3s 47ms/step - loss: 1.6584 - accuracy: 0.3990 - val_loss: 1.5267 - val_accuracy: 0.4676 - lr:
Epoch 4/20
56/56 [==============================] - 3s 48ms/step - loss: 1.5020 - accuracy: 0.4743 - val_loss: 1.4814 - val_accuracy: 0.4922 - lr:
Epoch 5/20
56/56 [==============================] - 3s 48ms/step - loss: 1.4813 - accuracy: 0.4771 - val_loss: 1.4404 - val_accuracy: 0.5257 - lr:
Epoch 6/20
56/56 [==============================] - 3s 47ms/step - loss: 1.3923 - accuracy: 0.5095 - val_loss: 1.4558 - val_accuracy: 0.5257 - lr:
Epoch 7/20
56/56 [==============================] - 3s 48ms/step - loss: 1.3263 - accuracy: 0.5352 - val_loss: 1.3458 - val_accuracy: 0.5503 - lr:
Epoch 8/20
56/56 [==============================] - 3s 51ms/step - loss: 1.2709 - accuracy: 0.5452 - val_loss: 1.5423 - val_accuracy: 0.5056 - lr:
Epoch 9/20
56/56 [==============================] - 3s 47ms/step - loss: 1.2431 - accuracy: 0.5575 - val_loss: 1.4280 - val_accuracy: 0.5145 - lr:
Epoch 10/20
56/56 [==============================] - ETA: 0s - loss: 1.2404 - accuracy: 0.5586
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
56/56 [==============================] - 3s 48ms/step - loss: 1.2404 - accuracy: 0.5586 - val_loss: 1.4301 - val_accuracy: 0.5459 - lr:
Epoch 11/20
56/56 [==============================] - 3s 48ms/step - loss: 1.1590 - accuracy: 0.5765 - val_loss: 1.3688 - val_accuracy: 0.5369 - lr:
Epoch 12/20
56/56 [==============================] - 3s 53ms/step - loss: 1.0858 - accuracy: 0.5971 - val_loss: 1.3709 - val_accuracy: 0.5414 - lr:
Epoch 13/20
56/56 [==============================] - ETA: 0s - loss: 1.0435 - accuracy: 0.6306
Epoch 13: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
56/56 [==============================] - 3s 48ms/step - loss: 1.0435 - accuracy: 0.6306 - val_loss: 1.5068 - val_accuracy: 0.4922 - lr:
Epoch 14/20
56/56 [==============================] - 3s 49ms/step - loss: 0.9786 - accuracy: 0.6484 - val_loss: 1.3861 - val_accuracy: 0.5548 - lr:
Epoch 15/20
56/56 [==============================] - 3s 48ms/step - loss: 0.9343 - accuracy: 0.6618 - val_loss: 1.4042 - val_accuracy: 0.5436 - lr:
Epoch 16/20
56/56 [==============================] - 3s 46ms/step - loss: 0.8774 - accuracy: 0.6847 - val_loss: 1.4881 - val_accuracy: 0.5615 - lr:
Epoch 17/20
56/56 [==============================] - 3s 46ms/step - loss: 0.8410 - accuracy: 0.6858 - val_loss: 1.4417 - val_accuracy: 0.5570 - lr:
Epoch 18/20
56/56 [==============================] - 3s 46ms/step - loss: 0.8126 - accuracy: 0.7026 - val_loss: 1.4338 - val_accuracy: 0.5660 - lr:
Epoch 19/20
56/56 [==============================] - 3s 46ms/step - loss: 0.7856 - accuracy: 0.7126 - val_loss: 1.4391 - val_accuracy: 0.5369 - lr:
Epoch 20/20
56/56 [==============================] - 3s 47ms/step - loss: 0.7386 - accuracy: 0.7355 - val_loss: 1.5912 - val_accuracy: 0.5548 - lr:
```

both training and validation accuracies are increased but still not good enough and it shows overfitting

accuracy: 0.7355 - val_loss: 1.5912 - val_accuracy: 0.5548 - lr: 2.5000e-04

Visualize the 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()
```


```
loss, accuracy = model.evaluate(train_ds, verbose=1,)
loss_v, accuracy_v = model.evaluate(val_ds, verbose=1)

print("Accuracy: ", accuracy)
print("Validation Accuracy: ",accuracy_v)
print("Loss: ",loss)
print("Validation Loss", loss_v)
```

```
56/56 [==============================] - 1s 19ms/step - loss: 0.6338 - accuracy: 0.7662
14/14 [==============================] - 0s 19ms/step - loss: 1.5912 - accuracy: 0.5548
Accuracy:  0.7661830186843872
Validation Accuracy:  0.5548098683357239
Loss:  0.6338330507278442
Validation Loss 1.5911773443222046
```

Model 1: Insights:

1. With the increase in the number of epochs training accuracy increased Final accuracy for training dataset is 76.6%
2. Validation accuracy is 55.48%
3. Training loss is .63
4. Validation loss is 1.52

We can see that validation dataset accuracy is significantly lower than training dataset accuracy, it is the sign of overfitting Overfitting happened since we have less number of training images and model has learnt from noise or unwanted details around the training data.

To address this issue we can use Data Augmentation to overcome overfitting.

Data augmentation will add additional training data by applying transformation on existing data. This will expose the model to more aspects of the data.

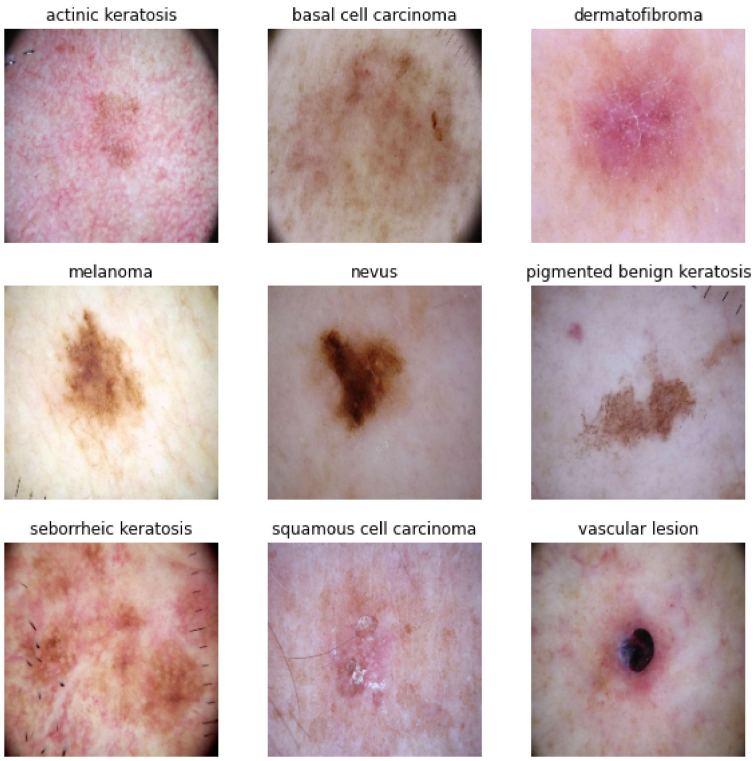
Model - 2: Model Building & training on the augmented data :

CNN Model with Augment strategy and Dropout layer

```
# Todo, after you have analysed the model fit history for presence of underfit or overfit, choose an appropriate data augumentation strategy.
# Your code goes here
with tf.device('/CPU:0'):
    data_augmentation = keras.Sequential(
      [
       # Random flip of image of horizontal axis
        layers.experimental.preprocessing.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
       # Random rotation of image
        layers.experimental.preprocessing.RandomRotation(0.2),
       # Random zoom of image
        layers.experimental.preprocessing.RandomZoom(0.2),
      ]
    )


# Todo, visualize how your augmentation strategy works for one instance of training image.
# Your code goes here
fig = plt.figure(figsize=(10, 10))

classes = range(len(class_names))
for images, labels in val_ds.take(2):
    missing = []
    for i in classes:
        ax = fig.add_subplot(3, 3, 1 + i)
        x = np.where(labels[:]==i)[0]
        if len(x) > 0:
            idx = np.where(labels[:]==i)[0][0]
            augmented_images = data_augmentation(images)
            im = images[idx].numpy().astype("uint8")
            plt.imshow(im)
            plt.title(class_names[labels[idx]])
            plt.axis("off")
        else:
            missing.append(i) # Appending list for missing classes
    if len(missing) > 0:
        classes = missing
        continue
    else:
        break
```



To do Create the model, compile and train the model

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

## Your code goes here

input_shape = (180,180,3)
lr = 1e-5
init = 'normal'
activ = 'relu'

model = Sequential()
#normalisation layer
model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(180, 180, 3)))
# Adding Conv layers
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu', padding ='same' ,input_shape=input_shape))
model.add(MaxPool2D(pool_size=(2, 2)))

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

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

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

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding ='same'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(9,kernel_regularizer=l2(0.01)))
model.add(Activation('softmax'))

## Number of classes is 9
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     (None, 180, 180, 3)       0

 module_wrapper_25 (ModuleWr  (None, 180, 180, 16)     448
 apper)

 module_wrapper_26 (ModuleWr  (None, 90, 90, 16)       0
 apper)

 module_wrapper_27 (ModuleWr  (None, 90, 90, 32)       4640
 apper)

 module_wrapper_28 (ModuleWr  (None, 45, 45, 32)       0
 apper)

 module_wrapper_29 (ModuleWr  (None, 45, 45, 64)       18496
 apper)

 module_wrapper_30 (ModuleWr  (None, 22, 22, 64)       0
 apper)

 module_wrapper_31 (ModuleWr  (None, 22, 22, 128)      73856
 apper)

 module_wrapper_32 (ModuleWr  (None, 11, 11, 128)      0
 apper)

 module_wrapper_33 (ModuleWr  (None, 11, 11, 256)      295168
 apper)

 module_wrapper_34 (ModuleWr  (None, 5, 5, 256)        0
 apper)

 module_wrapper_35 (ModuleWr  (None, 6400)             0
 apper)
```

```
 module_wrapper_36 (ModuleWr  (None, 512)              3277312
 apper)

 module_wrapper_37 (ModuleWr  (None, 512)              0
 apper)

 module_wrapper_38 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_39 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 3,674,537
Trainable params: 3,674,537
Non-trainable params: 0
_________________________________________________________________
```

Compile the model

```
## Your code goes here
#Choose an appropriate optimiser and loss function for model training
from tensorflow.keras.optimizers import Adam

optimizer = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
loss_function = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer,
              loss=loss_function,
              metrics=['accuracy'])
```

Training the model

```
## Your code goes here, note: train your model for 20 epochs

epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  callbacks=[learning_rate_reduction]
)
```

```
Epoch 1/20
56/56 [==============================] - 3s 32ms/step - loss: 2.0289 - accuracy: 0.2148 - val_loss: 1.8422 - val_accuracy: 0.3490 - lr:
Epoch 2/20
56/56 [==============================] - 2s 27ms/step - loss: 1.7938 - accuracy: 0.3376 - val_loss: 1.6914 - val_accuracy: 0.3691 - lr:
Epoch 3/20
56/56 [==============================] - 2s 27ms/step - loss: 1.7032 - accuracy: 0.3683 - val_loss: 1.6074 - val_accuracy: 0.4586 - lr:
Epoch 4/20
56/56 [==============================] - 2s 27ms/step - loss: 1.6076 - accuracy: 0.4291 - val_loss: 1.6709 - val_accuracy: 0.3982 - lr:
Epoch 5/20
56/56 [==============================] - 2s 27ms/step - loss: 1.5644 - accuracy: 0.4314 - val_loss: 1.6185 - val_accuracy: 0.4564 - lr:
Epoch 6/20
56/56 [==============================] - 1s 26ms/step - loss: 1.4635 - accuracy: 0.4916 - val_loss: 1.4247 - val_accuracy: 0.5056 - lr:
Epoch 7/20
56/56 [==============================] - 1s 26ms/step - loss: 1.4083 - accuracy: 0.5056 - val_loss: 1.4171 - val_accuracy: 0.4966 - lr:
Epoch 8/20
56/56 [==============================] - 1s 26ms/step - loss: 1.3702 - accuracy: 0.5201 - val_loss: 1.3884 - val_accuracy: 0.5190 - lr:
Epoch 9/20
56/56 [==============================] - 1s 26ms/step - loss: 1.3343 - accuracy: 0.5290 - val_loss: 1.4499 - val_accuracy: 0.4944 - lr:
Epoch 10/20
56/56 [==============================] - 1s 26ms/step - loss: 1.3255 - accuracy: 0.5301 - val_loss: 1.4906 - val_accuracy: 0.4877 - lr:
Epoch 11/20
56/56 [==============================] - 1s 26ms/step - loss: 1.2710 - accuracy: 0.5491 - val_loss: 1.4451 - val_accuracy: 0.5213 - lr:
Epoch 12/20
56/56 [==============================] - 1s 26ms/step - loss: 1.2766 - accuracy: 0.5513 - val_loss: 1.4433 - val_accuracy: 0.5235 - lr:
Epoch 13/20
56/56 [==============================] - 1s 26ms/step - loss: 1.2207 - accuracy: 0.5569 - val_loss: 1.3489 - val_accuracy: 0.5391 - lr:
Epoch 14/20
56/56 [==============================] - 1s 26ms/step - loss: 1.1841 - accuracy: 0.5675 - val_loss: 1.3669 - val_accuracy: 0.5145 - lr:
Epoch 15/20
56/56 [==============================] - 1s 26ms/step - loss: 1.1641 - accuracy: 0.5792 - val_loss: 1.4388 - val_accuracy: 0.5190 - lr:
Epoch 16/20
56/56 [==============================] - 1s 26ms/step - loss: 1.1345 - accuracy: 0.5882 - val_loss: 1.4122 - val_accuracy: 0.5459 - lr:
Epoch 17/20
```

```
56/56 [==============================] - 1s 26ms/step - loss: 1.0751 - accuracy: 0.6038 - val_loss: 1.4282 - val_accuracy: 0.5414 - lr:
Epoch 18/20
56/56 [==============================] - 1s 26ms/step - loss: 1.0783 - accuracy: 0.6138 - val_loss: 1.3725 - val_accuracy: 0.5436 - lr:
Epoch 19/20
55/56 [============================>.] - ETA: 0s - loss: 1.0182 - accuracy: 0.6318
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
56/56 [==============================] - 1s 26ms/step - loss: 1.0199 - accuracy: 0.6300 - val_loss: 1.3945 - val_accuracy: 0.5257 - lr:
Epoch 20/20
56/56 [==============================] - 1s 26ms/step - loss: 0.9225 - accuracy: 0.6641 - val_loss: 1.3822 - val_accuracy: 0.5705 - lr:
```

Visualizing the results

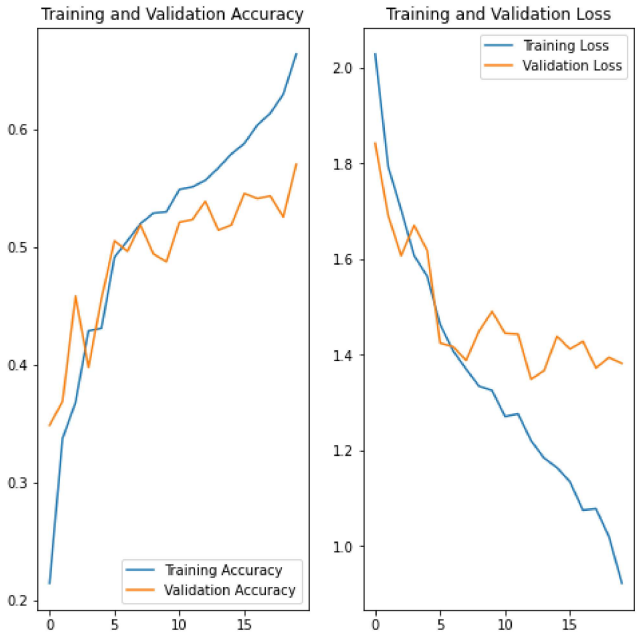
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



To do: Write your findings after the model fit, see if the earlier issue is resolved or not?

Model accuracies are: loss: 0.9225 - accuracy: 0.6641 - val_loss: 1.3822 - val_accuracy: 0.5705 - lr: 5.0000e-04

which shows the model is quite better in terms of address the overfitting but overall model accuaracies are significantly low..

It is not a good model as of now it sowhs underfitting need to improve the accuracies

```
## Your code goes here.

loss, accuracy = model.evaluate(train_ds, verbose=1,)
loss_v, accuracy_v = model.evaluate(val_ds, verbose=1)
```

```
print("Accuracy: ", accuracy)
print("Validation Accuracy: ",accuracy_v)
print("Loss: ",loss)
print("Validation Loss", loss_v)
```

```
56/56 [==============================] - 1s 14ms/step - loss: 0.7950 - accuracy: 0.7115
14/14 [==============================] - 0s 12ms/step - loss: 1.3822 - accuracy: 0.5705
Accuracy:  0.7114955186843872
Validation Accuracy:  0.5704697966575623
Loss:  0.7949563264846802
Validation Loss 1.3821607828140259
```

- both training and validation accuracies got increased due to increase in epochs.
- Training accuracies and validation accuracies are increased.
- Model is underfitting -- validation accuracy is decreased to 57% from 71%
- still need to improve the model

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

```
Collaboration/Train/melanoma/ISIC_0010962.jpg',
 '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging
```

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

```
 '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',
 ...]
```

Another reason of low performance of model could be the immbalance data:

Lets check that:

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

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

```
dermatofibroma                95
seborrheic keratosis          77
Name: Label, dtype: int64
```

There is clear sign of class imbalance in the dataset:

1. seborrheic keratosis and dermatofibroma are having least images.
2. pigmented benign keratosis and melanoma are more
3. Need to address the class imbalance to address underfitting

Handling class imbalances:

Rectify class imbalances present in the training dataset with Augmentor library.

```
!pip install Augmentor
```

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

```
path_to_training_dataset=str(data_dir_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) ## 500 samples are added to all the categories of images to make it more uniform.
```

```
noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/actinic keratosis/output.Processing <PIL.Image.I

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/basal cell carcinoma/output.Processing <PIL.Imag

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/dermatofibroma/output.Processing <PIL.Image.Imag

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/melanoma/output.Processing <PIL.Image.Image imag

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/nevus/output.Processing <PIL.Image.Image image m

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/pigmented benign keratosis/output.Processing <PI

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/seborrheic keratosis/output.Processing <PIL.Imag

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous cell carcinoma/output.Processing <PIL.I

noma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular lesion/output.Processing <PIL.JpegImage
```

```
# Total count of augmented image
```

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

```
4500
```

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
```

```
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0029533.jpg_35443d2e-591f-4fb5-bb72-16816774a943.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0030280.jpg_f44da450-1efa-4bc6-bd7d-c88c64796291.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0031672.jpg_8dd5cecd-1a4c-42ad-9a4b-acb9fc426df7.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0027708.jpg_603f2e74-4d48-4739-827f-fe66b6c20f96.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0028132.jpg_bb8922cf-0c93-4855-85c4-e0e76e21b8ea.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0032455.jpg_8d94795e-5300-4bc8-8238-80c8a6e12910.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0027303.jpg_0e4073c8-b1d8-47c1-b238-9ce0aacab60f.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0026981.jpg_da474239-9f58-45e2-bd1a-0d432646215d.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0030375.jpg_51b3e7c8-70a8-4e0c-8718-42de0fc6c79d.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0032455.jpg_8cfbcd76-253f-4b8e-997a-3f2f6bcc589c.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0028224.jpg_64a500aa-6af9-47bd-a00b-4da956f61dbc.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0032356.jpg_dfe3ad9e-8b57-4156-a46e-bb10ab8f9814.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0028795.jpg_b9825695-2388-4007-b9d5-5f27c9afb02a.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0032329.jpg_75d1824d-c31a-4ce9-8f09-3e74a8a77957.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0026927.jpg_2a783ac1-d9be-486d-a326-e3d0039682b8.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0024946.jpg_164eb100-f9e7-49fc-b7f9-1961564fed5a.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0029567.jpg_397cf7fe-ebc2-4e11-bbdf-4ad21893c7e1.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0024946.jpg_3c769ec7-59ba-448c-8942-b7522032e836.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0029533.jpg_09b2a427-8365-4562-bc4b-36006460c0f7.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0027767.jpg_746b53be-49e2-4987-8c45-725f7cf6da25.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0031211.jpg_c60ee827-bfaa-48ac-931d-813d2ad22256.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0025130.jpg_4767f054-7c00-429d-bc15-75d5a4cf1dd2.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0025712.jpg_b6f4c123-a4a9-4320-8fb6-7dff578f5d25.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0026720.jpg_595057e0-187e-444f-a1d7-8f9e80ede142.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0031692.jpg_3387e811-b341-4257-92be-2ec6ffa4d3cf.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0032154.jpg_e9559cec-9e10-46a4-9412-3adb4bec5751.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0031852.jpg_300ea0d8-050f-4595-83df-70732eb2517b.jpg',
        '/content/gdrive/MyDrive/CNN_Melanoma_Detection_sukh/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous
       cell carcinoma/output/squamous cell carcinoma_original_ISIC_0028816.jpg_814797a6-2419-46be-8a41-ca5028a9c818.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
```

```
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 'squamous cell carcinoma',
 ...]
```

```
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)
```

```
#value counts after augmentation

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
```

The number of images in each categories are more now which would solve the problem

Train the model on the data created using Augmentor

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

Train data

```
#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", ## Todo choose the correct parameter value, so that only training data is refered to,,
```

```
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

Validation data set creation

```
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 data is refered to,
  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)

Model 3: CNN Model with Augmentor to rectify imbalance class data

```
input_shape = (180,180,3)
lr = 1e-5
init = 'normal'
activ = 'relu'

model = Sequential()
#normalisation layer
model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255, input_shape=(180, 180, 3)))
# Adding Conv layer
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu', padding ='same' ,input_shape=input_shape))
model.add(MaxPool2D(pool_size=(2, 2)))

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

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

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

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding ='same'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(512, activation='relu'))

model.add(Dense(9,kernel_regularizer=l2(0.01)))
model.add(Activation('softmax'))

## Number of classes is 9
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_4 (Rescaling)     (None, 180, 180, 3)       0

 module_wrapper_40 (ModuleWr  (None, 180, 180, 16)     448
 apper)

 module_wrapper_41 (ModuleWr  (None, 90, 90, 16)       0
 apper)

 module_wrapper_42 (ModuleWr  (None, 90, 90, 32)       4640
 apper)

 module_wrapper_43 (ModuleWr  (None, 45, 45, 32)       0
 apper)
```

```
 module_wrapper_44 (ModuleWr  (None, 45, 45, 64)       18496
 apper)

 module_wrapper_45 (ModuleWr  (None, 22, 22, 64)       0
 apper)

 module_wrapper_46 (ModuleWr  (None, 22, 22, 128)      73856
 apper)

 module_wrapper_47 (ModuleWr  (None, 11, 11, 128)      0
 apper)

 module_wrapper_48 (ModuleWr  (None, 11, 11, 256)      295168
 apper)

 module_wrapper_49 (ModuleWr  (None, 5, 5, 256)        0
 apper)

 module_wrapper_50 (ModuleWr  (None, 5, 5, 256)        0
 apper)

 module_wrapper_51 (ModuleWr  (None, 6400)             0
 apper)

 module_wrapper_52 (ModuleWr  (None, 512)              3277312
 apper)

 module_wrapper_53 (ModuleWr  (None, 9)                4617
 apper)

 module_wrapper_54 (ModuleWr  (None, 9)                0
 apper)

=================================================================
Total params: 3,674,537
Trainable params: 3,674,537
Non-trainable params: 0
_________________________________________________________________
```

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

```
### choosing an appropirate optimiser and loss function
optimizer = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
loss_function = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer,
              loss=loss_function,
              metrics=['accuracy'])
```

Train the model for 20 epochs

```
epochs = 30
## Your code goes here, use 30 epochs.
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  callbacks=[learning_rate_reduction]
)
```

```
Epoch 2/30
169/169 [==============================] - 33s 188ms/step - loss: 1.5395 - accuracy: 0.3993 - val_loss: 1.3560 - val_accuracy: 0.4610
Epoch 3/30
169/169 [==============================] - 32s 185ms/step - loss: 1.3891 - accuracy: 0.4607 - val_loss: 1.3132 - val_accuracy: 0.4618
Epoch 4/30
169/169 [==============================] - 31s 180ms/step - loss: 1.2702 - accuracy: 0.5078 - val_loss: 1.2117 - val_accuracy: 0.5241
Epoch 5/30
169/169 [==============================] - 31s 180ms/step - loss: 1.1142 - accuracy: 0.5603 - val_loss: 1.1264 - val_accuracy: 0.5813
Epoch 6/30
169/169 [==============================] - 31s 179ms/step - loss: 0.9929 - accuracy: 0.6122 - val_loss: 1.1317 - val_accuracy: 0.5664
Epoch 7/30
169/169 [==============================] - 31s 179ms/step - loss: 0.8205 - accuracy: 0.6877 - val_loss: 0.8980 - val_accuracy: 0.6711
Epoch 8/30
169/169 [==============================] - 33s 189ms/step - loss: 0.7408 - accuracy: 0.7166 - val_loss: 0.8156 - val_accuracy: 0.6919
Epoch 9/30
169/169 [==============================] - 31s 180ms/step - loss: 0.5929 - accuracy: 0.7778 - val_loss: 0.9021 - val_accuracy: 0.6763
```

```
169/169 [==============================] - 32s 182ms/step - loss: 0.3894 - accuracy: 0.8557 - val_loss: 0.6474 - val_accuracy: 0.7929
Epoch 12/30
169/169 [==============================] - 31s 179ms/step - loss: 0.3390 - accuracy: 0.8661 - val_loss: 0.7187 - val_accuracy: 0.7862
Epoch 13/30
169/169 [==============================] - 32s 182ms/step - loss: 0.3015 - accuracy: 0.8815 - val_loss: 0.7099 - val_accuracy: 0.7921
Epoch 14/30
169/169 [==============================] - 32s 187ms/step - loss: 0.2608 - accuracy: 0.9002 - val_loss: 0.6041 - val_accuracy: 0.8285
Epoch 15/30
169/169 [==============================] - 31s 180ms/step - loss: 0.2145 - accuracy: 0.9152 - val_loss: 0.6341 - val_accuracy: 0.8233
Epoch 16/30
169/169 [==============================] - 32s 181ms/step - loss: 0.2081 - accuracy: 0.9203 - val_loss: 0.6337 - val_accuracy: 0.8434
Epoch 17/30
169/169 [==============================] - 32s 182ms/step - loss: 0.1944 - accuracy: 0.9217 - val_loss: 0.5942 - val_accuracy: 0.8523
Epoch 18/30
169/169 [==============================] - 31s 180ms/step - loss: 0.1830 - accuracy: 0.9260 - val_loss: 0.7779 - val_accuracy: 0.8211
Epoch 19/30
169/169 [==============================] - 31s 179ms/step - loss: 0.1946 - accuracy: 0.9217 - val_loss: 0.5990 - val_accuracy: 0.8582
Epoch 20/30
169/169 [==============================] - 33s 189ms/step - loss: 0.1703 - accuracy: 0.9318 - val_loss: 0.6689 - val_accuracy: 0.8448
Epoch 21/30
169/169 [==============================] - 31s 179ms/step - loss: 0.1899 - accuracy: 0.9303 - val_loss: 0.5897 - val_accuracy: 0.8508
Epoch 22/30
169/169 [==============================] - 32s 181ms/step - loss: 0.1856 - accuracy: 0.9301 - val_loss: 0.6094 - val_accuracy: 0.8641
Epoch 23/30
169/169 [==============================] - 31s 179ms/step - loss: 0.1592 - accuracy: 0.9349 - val_loss: 0.5849 - val_accuracy: 0.8537
Epoch 24/30
169/169 [==============================] - 31s 179ms/step - loss: 0.1710 - accuracy: 0.9321 - val_loss: 0.6252 - val_accuracy: 0.8649
Epoch 25/30
169/169 [==============================] - 31s 179ms/step - loss: 0.1562 - accuracy: 0.9319 - val_loss: 0.6270 - val_accuracy: 0.8434
Epoch 26/30
169/169 [==============================] - 33s 188ms/step - loss: 0.2021 - accuracy: 0.9280 - val_loss: 0.5474 - val_accuracy: 0.8634
Epoch 27/30
169/169 [==============================] - 31s 180ms/step - loss: 0.1245 - accuracy: 0.9473 - val_loss: 0.6858 - val_accuracy: 0.8671
Epoch 28/30
169/169 [==============================] - 31s 178ms/step - loss: 0.1404 - accuracy: 0.9425 - val_loss: 0.6481 - val_accuracy: 0.8560
Epoch 29/30
169/169 [==============================] - 32s 181ms/step - loss: 0.1202 - accuracy: 0.9481 - val_loss: 0.6604 - val_accuracy: 0.8627
Epoch 30/30
```

```
loss, accuracy = model.evaluate(train_ds, verbose=1,)
loss_v, accuracy_v = model.evaluate(val_ds, verbose=1)

print("Accuracy: ", accuracy)
print("Validation Accuracy: ",accuracy_v)
print("Loss: ",loss)
print("Validation Loss", loss_v)
```

```
169/169 [==============================] - 24s 137ms/step - loss: 0.0974 - accuracy: 0.9603
43/43 [==============================] - 6s 113ms/step - loss: 0.5643 - accuracy: 0.8716
Accuracy:  0.9603115916252136
Validation Accuracy:  0.8715664148330688
Loss:  0.09742846339941025
Validation Loss 0.5643184781074524
```

To do: 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()
```

Insight of Final Model

Training accuracy is 95.51% and validation accuracy is 87.3 %

After using Augmentor and using imbalance of all classes, significance improvement in the training and validation accuracy are observed.

More epochs can be added to increase the accuracy Further but too much epoch would cause overfitting too. so epoch =30 is quite great enough

Overfitting or underfitting issues are resloved in the final model

Training and validation loss has been decreased due to increase in epoch