```
import os
import PIL
import pathlib
import pandas as pd
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import RMSprop
```

```
''' train and test path '''
train_path = pathlib.Path("../input/skin-cancer/Skin cancer ISIC The International Skin Imaging Collaboration/Train")
test_path = pathlib.Path('../input/skin-cancer/Skin cancer ISIC The International Skin Imaging Collaboration/Test')
```

```
''' length of train data '''
train_image_len = len(list(train_path.glob('*/*.jpg')))
print(train_image_len)

''' length of test data '''
test_image_len = len(list(test_path.glob('*/*.jpg')))
print(test_image_len)
```

```
2239
118
```

```
''' initializing some variables '''
batch_size = 32
img_h = 180
img_w = 180
```

```
''' data augmentation '''
''' using 80-20 split of data, 80% for training and 20% for validation '''
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_path,
    seed=123,
    validation_split= 0.2,
    subset= 'training',
    image_size=(img_h, img_w),
    batch_size = batch_size)
```

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

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_path,
    seed=123,
    validation_split= 0.2,
    subset= 'validation',
    image_size=(img_h, img_w),
    batch_size = batch_size
)
```

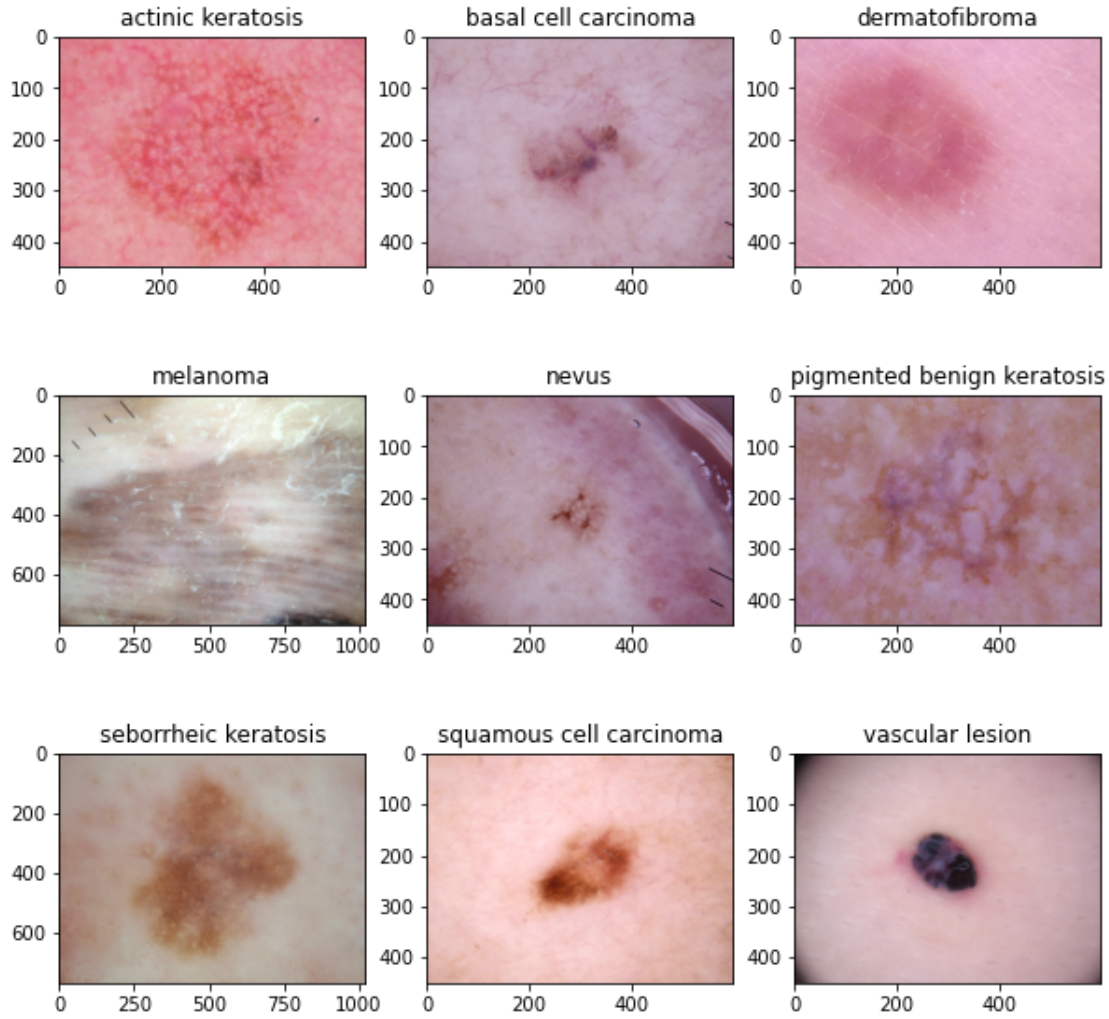
```
Found 2239 files belonging to 9 classes.
Using 447 files for validation.
```

```
''' classes names '''
c_names = train_ds.class_names
print(c_names)
```

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

```
''' plotting some images '''
for i in range(9):
    plt.figure(figsize=(10,10))
    plt.subplot(3, 3, i + 1)
```

```
    img = mpimg.imread(str(list(train_ds.glob(c_names[i]+'/*.jpg'))[1]))
    plt.title(c_names[i])
    plt.imshow(image)
```




```
''' AUTOTUNE '''
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 '''
n_c = 9

''' preprocessing layer '''
model = Sequential([layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_h, img_,3))])

''' Convolutional Layers '''
model.add(Conv2D(filters=32, kernel_size=(5,5), padding='Same', activation ='relu', input_shape = (180, 180, 32)))
model.add(Conv2D(filters=32, kernel_size=(5,5), padding = 'Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(5,5), padding='Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(5,5), padding='Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(5,5),padding='Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

''' adding dropout '''
model.add(Dropout(0.25))
model.add(Flatten())

''' Classification Layer'''
model.add(Dense(n_c, activation = "softmax"))
```

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

```
''' lets see how looks like '''
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
rescaling (Rescaling)        (None, 180, 180, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 180, 180, 32)      2432
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 180, 180, 32)      25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 90, 90, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 90, 90, 32)        25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 45, 45, 32)        25632
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 22, 22, 32)        25632
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 11, 11, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 3872)              0
_________________________________________________________________
dense (Dense)                (None, 9)                 34857
=================================================================
Total params: 139,817
Trainable params: 139,817
Non-trainable params: 0
_________________________________________________________________
```

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

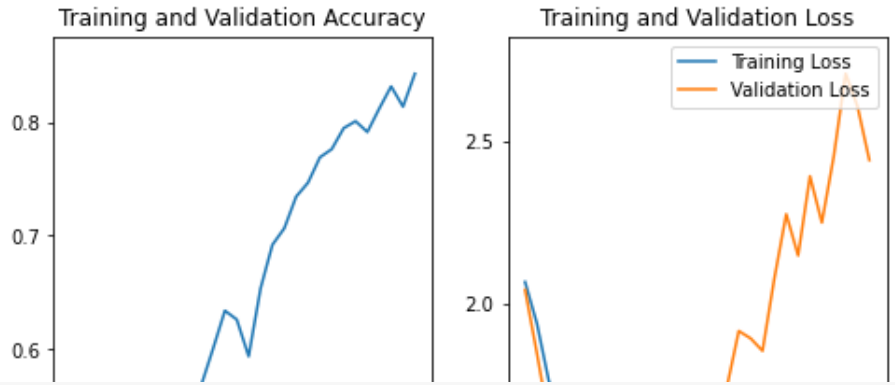
```
Epoch 1/30
56/56 [==============================] - 39s 187ms/step - loss: 2.0967 - accuracy: 0.1792 - val_loss: 2.0419 - val_accuracy: 0.2260
Epoch 2/30
56/56 [==============================] - 2s 42ms/step - loss: 1.9621 - accuracy: 0.2729 - val_loss: 1.8453 - val_accuracy: 0.3221
Epoch 3/30
56/56 [==============================] - 2s 42ms/step - loss: 1.7887 - accuracy: 0.3274 - val_loss: 1.6378 - val_accuracy: 0.3915
```

```
Epoch 4/30
56/56 [==============================] - 2s 43ms/step - loss: 1.6739 - accuracy: 0.3888 - val_loss: 1.6628 - val_accuracy: 0.4340
Epoch 5/30
56/56 [==============================] - 2s 42ms/step - loss: 1.6105 - accuracy: 0.4201 - val_loss: 1.6095 - val_accuracy: 0.4228
Epoch 6/30
56/56 [==============================] - 2s 42ms/step - loss: 1.5925 - accuracy: 0.4263 - val_loss: 1.6190 - val_accuracy: 0.4564
Epoch 7/30
56/56 [==============================] - 2s 42ms/step - loss: 1.5462 - accuracy: 0.4509 - val_loss: 1.5661 - val_accuracy: 0.4362
Epoch 8/30
56/56 [==============================] - 2s 42ms/step - loss: 1.4550 - accuracy: 0.4695 - val_loss: 1.5336 - val_accuracy: 0.4698
Epoch 9/30
56/56 [==============================] - 2s 42ms/step - loss: 1.4269 - accuracy: 0.5035 - val_loss: 1.5417 - val_accuracy: 0.4743
Epoch 10/30
56/56 [==============================] - 2s 42ms/step - loss: 1.3674 - accuracy: 0.5168 - val_loss: 1.5359 - val_accuracy: 0.4452
Epoch 11/30
56/56 [==============================] - 2s 41ms/step - loss: 1.2223 - accuracy: 0.5780 - val_loss: 1.5020 - val_accuracy: 0.5190
Epoch 12/30
56/56 [==============================] - 2s 42ms/step - loss: 1.1950 - accuracy: 0.5645 - val_loss: 1.5391 - val_accuracy: 0.5235
Epoch 13/30
56/56 [==============================] - 2s 43ms/step - loss: 1.1311 - accuracy: 0.5892 - val_loss: 1.6529 - val_accuracy: 0.5235
Epoch 14/30
56/56 [==============================] - 2s 43ms/step - loss: 1.0636 - accuracy: 0.6261 - val_loss: 1.7371 - val_accuracy: 0.5235
Epoch 15/30
56/56 [==============================] - 2s 42ms/step - loss: 1.0413 - accuracy: 0.6229 - val_loss: 1.5508 - val_accuracy: 0.4944
Epoch 16/30
56/56 [==============================] - 2s 42ms/step - loss: 1.0484 - accuracy: 0.6232 - val_loss: 1.5600 - val_accuracy: 0.5034
Epoch 17/30
56/56 [==============================] - 2s 41ms/step - loss: 0.9400 - accuracy: 0.6587 - val_loss: 1.7019 - val_accuracy: 0.5213
Epoch 18/30
56/56 [==============================] - 2s 42ms/step - loss: 0.8309 - accuracy: 0.6997 - val_loss: 1.7408 - val_accuracy: 0.5078
Epoch 19/30
56/56 [==============================] - 2s 41ms/step - loss: 0.7784 - accuracy: 0.7047 - val_loss: 1.9144 - val_accuracy: 0.4944
Epoch 20/30
56/56 [==============================] - 2s 41ms/step - loss: 0.6429 - accuracy: 0.7519 - val_loss: 1.8915 - val_accuracy: 0.4944
Epoch 21/30
56/56 [==============================] - 2s 44ms/step - loss: 0.6920 - accuracy: 0.7414 - val_loss: 1.8530 - val_accuracy: 0.5034
Epoch 22/30
56/56 [==============================] - 2s 41ms/step - loss: 0.6488 - accuracy: 0.7587 - val_loss: 2.0778 - val_accuracy: 0.4899
Epoch 23/30
56/56 [==============================] - 2s 42ms/step - loss: 0.5768 - accuracy: 0.7907 - val_loss: 2.2759 - val_accuracy: 0.4676
Epoch 24/30
56/56 [==============================] - 2s 41ms/step - loss: 0.4917 - accuracy: 0.8125 - val_loss: 2.1480 - val_accuracy: 0.4855
Epoch 25/30
56/56 [==============================] - 2s 41ms/step - loss: 0.4831 - accuracy: 0.8069 - val_loss: 2.3934 - val_accuracy: 0.4497
```

```
Epoch 26/30
56/56 [==============================] - 2s 41ms/step - loss: 0.5141 - accuracy: 0.8212 - val_loss: 2.2497 - val_accuracy: 0.4318
Epoch 27/30
56/56 [==============================] - 2s 42ms/step - loss: 0.4883 - accuracy: 0.8206 - val_loss: 2.4547 - val_accuracy: 0.5011
Epoch 28/30
56/56 [==============================] - 2s 42ms/step - loss: 0.4038 - accuracy: 0.8429 - val_loss: 2.7105 - val_accuracy: 0.4966
Epoch 29/30
```

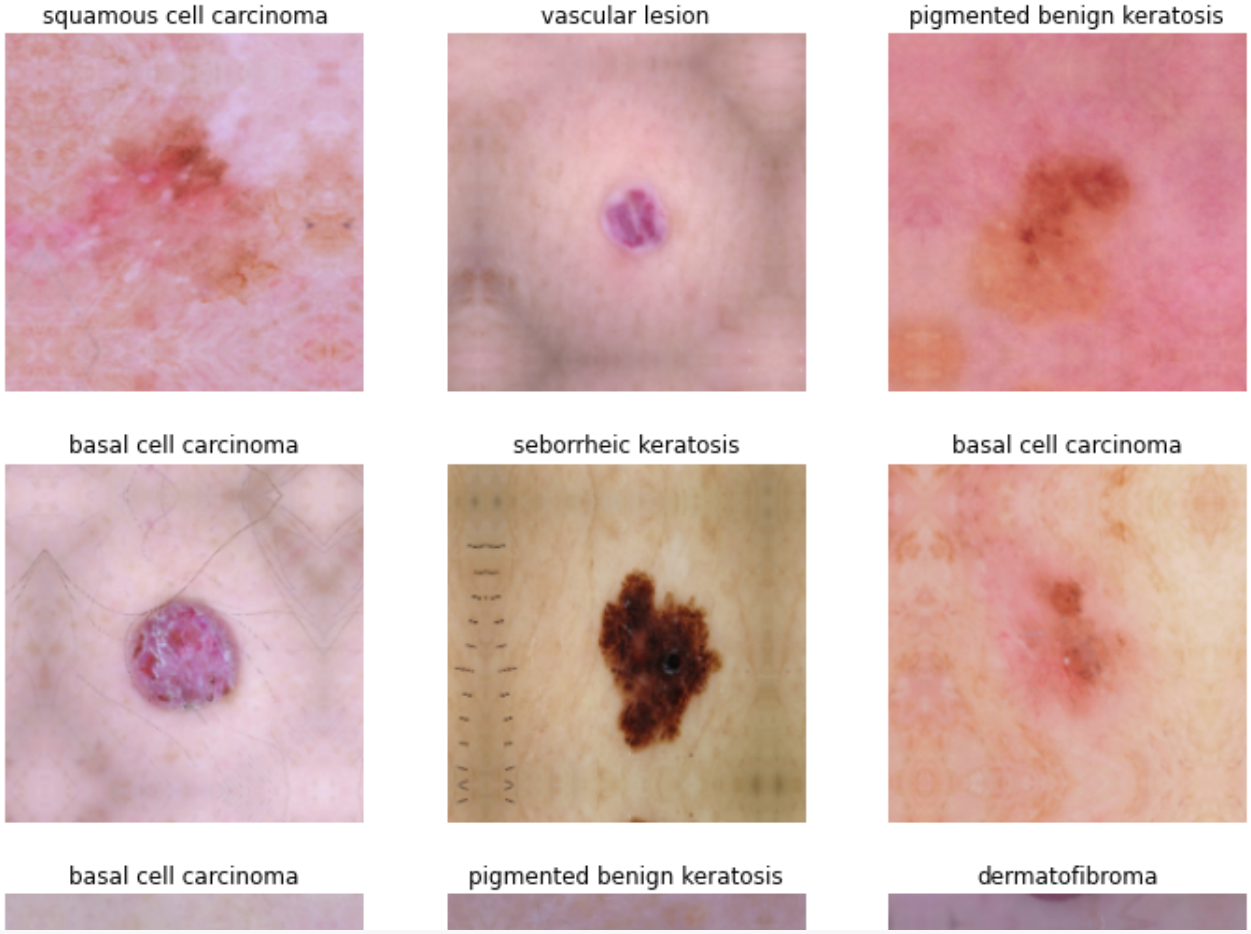
```
''' plotting training accuracy and validation accuracy graph '''
epochs_range = range(epochs)
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

''' plotting training loss and validation loss graph '''
plt.subplot(1, 2, 2)
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
plt.plot(epochs_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




```
''' augmentation '''
augmentation = keras.Sequential([
                            layers.experimental.preprocessing.RandomFlip(mode="horizontal_and_vertical",
                                                                         input_shape=(img_height,img_width,3)),
                            layers.experimental.preprocessing.RandomRotation(0.2, fill_mode='reflect'),
                            layers.experimental.preprocessing.RandomZoom(height_factor=(0.2, 0.3),
                                                                         width_factor=(0.2, 0.3), fill_mode='reflect')])
```

```
''' ploting some augmented images '''
for img, lbls in train_ds.take(1):
    plt.figure(figsize=(12, 12))
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmentation(img)[i].numpy().astype("uint8"))
        plt.title(c_names[lbl[i]])
        plt.axis("off")
```




```
''' Model with Augmentation '''
n_classes = 9

''' preprocess layer '''
model = Sequential([augmentation, layers.experimental.preprocessing.Rescaling(1./255,
                                                                               input_shape=(img_height, img_width,3))])
''' Convolutional layers '''
model.add(Conv2D(filters = 32, kernel_size=(5,5), padding='Same',  activation ='relu', input_shape=(180, 180, 32)))
model.add(Conv2D(filters = 32, kernel_size=(5,5),padding='Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(5,5), padding='Same', activation ='relu'))
```

```
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(5,5), padding='Same', activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

''' adding dropout '''
model.add(Dropout(0.25))
model.add(Flatten())

''' classification layer '''
model.add(Dense(n_classes, activation = "softmax"))

```

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

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

```
Epoch 1/30
56/56 [==============================] - 3s 46ms/step - loss: 2.0556 - accuracy: 0.2013 - val_loss: 1.8614 - val_accuracy: 0.3221
Epoch 2/30
56/56 [==============================] - 2s 43ms/step - loss: 1.8633 - accuracy: 0.3309 - val_loss: 1.6795 - val_accuracy: 0.4228
Epoch 3/30
56/56 [==============================] - 2s 43ms/step - loss: 1.6612 - accuracy: 0.4078 - val_loss: 1.5704 - val_accuracy: 0.4430
Epoch 4/30
56/56 [==============================] - 2s 44ms/step - loss: 1.5722 - accuracy: 0.4323 - val_loss: 1.4883 - val_accuracy: 0.4855
Epoch 5/30
56/56 [==============================] - 2s 43ms/step - loss: 1.5263 - accuracy: 0.4476 - val_loss: 1.6166 - val_accuracy: 0.4586
Epoch 6/30
56/56 [==============================] - 2s 43ms/step - loss: 1.6074 - accuracy: 0.4138 - val_loss: 1.5532 - val_accuracy: 0.4787
Epoch 7/30
56/56 [==============================] - 2s 44ms/step - loss: 1.5026 - accuracy: 0.4825 - val_loss: 1.5627 - val_accuracy: 0.4519
Epoch 8/30
56/56 [==============================] - 2s 44ms/step - loss: 1.4693 - accuracy: 0.4720 - val_loss: 1.4823 - val_accuracy: 0.5190
Epoch 9/30
56/56 [==============================] - 2s 43ms/step - loss: 1.4470 - accuracy: 0.4982 - val_loss: 1.4500 - val_accuracy: 0.5168
```

```
Epoch 10/30
56/56 [==============================] - 2s 43ms/step - loss: 1.4310 - accuracy: 0.4975 - val_loss: 1.4505 - val_accuracy: 0.4765
Epoch 11/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3865 - accuracy: 0.4889 - val_loss: 1.4905 - val_accuracy: 0.4944
Epoch 12/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3883 - accuracy: 0.5242 - val_loss: 1.4624 - val_accuracy: 0.5302
Epoch 13/30
56/56 [==============================] - 3s 45ms/step - loss: 1.3312 - accuracy: 0.5172 - val_loss: 1.4012 - val_accuracy: 0.5011
Epoch 14/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3640 - accuracy: 0.5344 - val_loss: 1.4617 - val_accuracy: 0.5235
Epoch 15/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3366 - accuracy: 0.5079 - val_loss: 1.5188 - val_accuracy: 0.5011
Epoch 16/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3909 - accuracy: 0.5090 - val_loss: 1.3510 - val_accuracy: 0.5436
Epoch 17/30
56/56 [==============================] - 2s 44ms/step - loss: 1.2923 - accuracy: 0.5403 - val_loss: 1.4794 - val_accuracy: 0.5056
Epoch 18/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3459 - accuracy: 0.5188 - val_loss: 1.3465 - val_accuracy: 0.5302
Epoch 19/30
56/56 [==============================] - 2s 44ms/step - loss: 1.2748 - accuracy: 0.5525 - val_loss: 1.3782 - val_accuracy: 0.5414
Epoch 20/30
56/56 [==============================] - 2s 44ms/step - loss: 1.2890 - accuracy: 0.5380 - val_loss: 1.3602 - val_accuracy: 0.5235
Epoch 21/30
56/56 [==============================] - 2s 44ms/step - loss: 1.3066 - accuracy: 0.5302 - val_loss: 1.3779 - val_accuracy: 0.4810
Epoch 22/30
56/56 [==============================] - 2s 44ms/step - loss: 1.3548 - accuracy: 0.5047 - val_loss: 1.3801 - val_accuracy: 0.5168
Epoch 23/30
56/56 [==============================] - 2s 43ms/step - loss: 1.2826 - accuracy: 0.5523 - val_loss: 1.4509 - val_accuracy: 0.4966
Epoch 24/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3252 - accuracy: 0.5270 - val_loss: 1.4411 - val_accuracy: 0.5078
Epoch 25/30
56/56 [==============================] - 2s 44ms/step - loss: 1.2554 - accuracy: 0.5508 - val_loss: 1.3420 - val_accuracy: 0.5190
Epoch 26/30
56/56 [==============================] - 2s 44ms/step - loss: 1.2414 - accuracy: 0.5648 - val_loss: 1.4564 - val_accuracy: 0.4944
Epoch 27/30
56/56 [==============================] - 2s 43ms/step - loss: 1.3913 - accuracy: 0.4851 - val_loss: 1.3729 - val_accuracy: 0.5280
Epoch 28/30
56/56 [==============================] - 2s 43ms/step - loss: 1.2726 - accuracy: 0.5415 - val_loss: 1.3174 - val_accuracy: 0.5235
Epoch 29/30
```

```
''' plotting training accuracy and validation accuracy graph '''
epochs_range = range(epochs)
plt.figure(figsize=(8, 8))
```

```
plt.subplot(1, 2, 1)
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

''' plotting training loss and validation loss graph '''
plt.subplot(1, 2, 2)
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
plt.plot(epochs_range, history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
p_list=[]
l_list=[]
for i in c_names:
    for j in train_path.glob(i+'/*.jpg'):
        p_list.append(str(j))
        l_list.append(i)

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

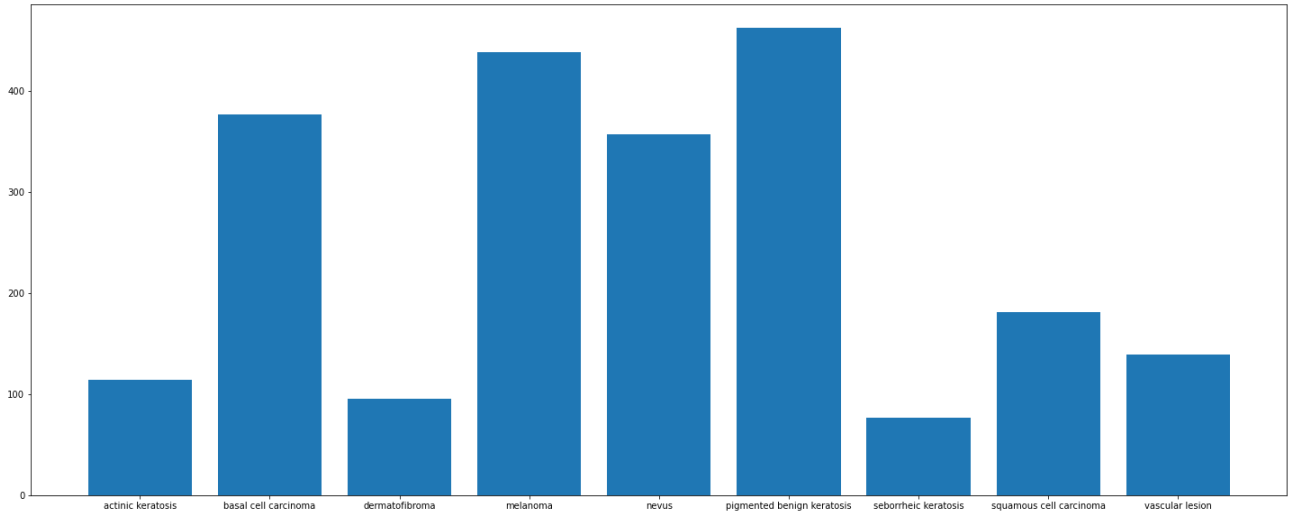
	Path	Label
0	../input/skin-cancer/Skin cancer ISIC The Inte...	actinic keratosis
1	../input/skin-cancer/Skin cancer ISIC The Inte...	actinic keratosis
2	../input/skin-cancer/Skin cancer ISIC The Inte...	actinic keratosis
3	../input/skin-cancer/Skin cancer ISIC The Inte...	actinic keratosis
4	../input/skin-cancer/Skin cancer ISIC The Inte...	actinic keratosis
...	...	...
2234	../input/skin-cancer/Skin cancer ISIC The Inte...	vascular lesion
2235	../input/skin-cancer/Skin cancer ISIC The Inte...	vascular lesion
2236	../input/skin-cancer/Skin cancer ISIC The Inte...	vascular lesion
2237	../input/skin-cancer/Skin cancer ISIC The Inte...	vascular lesion
2238	../input/skin-cancer/Skin cancer ISIC The Inte...	vascular lesion

2239 rows × 2 columns

```
''' barplot '''
count=[]
for i in c_names:
    count.append(len(list(train_path.glob(i+'/*.jpg'))))
```

```
plt.figure(figsize=(25,10))
plt.bar(class_names,count)
```

```
<BarContainer object of 9 artists>
```




```
1
```