Problem Statement

To build a CNN based model which can accurately detect melanoma. Melanoma is a type of cancer that can be deadly if not detected early. It accounts for 75% of skin cancer deaths. A solution which can evaluate images and alert the dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

Load Python Library

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

Load Dataset

```
#connect to gdrive and mount it
from google.colab import drive
drive.mount("/content/gdrive")
```

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

```
# get the train & test folder path location in gdrive
data_dir_train = pathlib.Path("/content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin can
data_dir_test  = pathlib.Path("/content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin can
```

```
#list directory in train folder
dir_train = os.listdir(data_dir_train)
dir_train.sort()
dir_train
```

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

```
#list dir in test folder
dir_test = os.listdir(data_dir_test)
dir_test.sort()
dir_test
```

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

```
#both test & train have same folders (disease folder ), now check the no. of datapoints in

#total train dataset
total_train_data = len(list(data_dir_train.glob("*/*.jpg")))
total_train_data
```

```
2239
```

```
#total test dataset
total_test_data = len(list(data_dir_test.glob("*/*.jpg")))
total_test_data
```

```
118
```

```
# train data in each folders
data_detail_pd = pd.DataFrame(columns=["Dir_Name","Total Image(Train)","Total Percentage(T

for dir_name in dir_train:
   total_image_in_folder = len(list(data_dir_train.glob(dir_name+"/*.jpg")))
   df = {"Dir_Name":dir_name,"Total Image(Train)":total_image_in_folder,"Total Percentage(
   data_detail_pd = data_detail_pd.append(df,ignore_index=True)

data_detail_pd = data_detail_pd.set_index("Dir_Name")
#display(data_detail_pd.sort_values(by="Total Percentage(Train)",ascending=False))


# test data in each folders

for dir_name in dir_test:
   total_image_in_folder = len(list(data_dir_test.glob(dir_name+"/*.jpg")))
   data_detail_pd.loc[dir_name,"Total Image(Test)"]  = total_image_in_folder
   data_detail_pd.loc[dir_name,"Total Percentage(Test)"]  = round((total_image_in_folder/t
display(data_detail_pd.sort_values(by="Total Percentage(Train)",ascending=False))
```

Dir_Name	Total Image(Train)	Total Percentage(Train)	Total Image(Test)	Total Percentage(Test)
pigmented benign keratosis	462	20.63	16.0	0.71
melanoma	438	19.56	16.0	0.71
basal cell carcinoma	376	16.79	16.0	0.71
nevus	357	15.94	16.0	0.71
squamous cell carcinoma	181	8.08	16.0	0.71

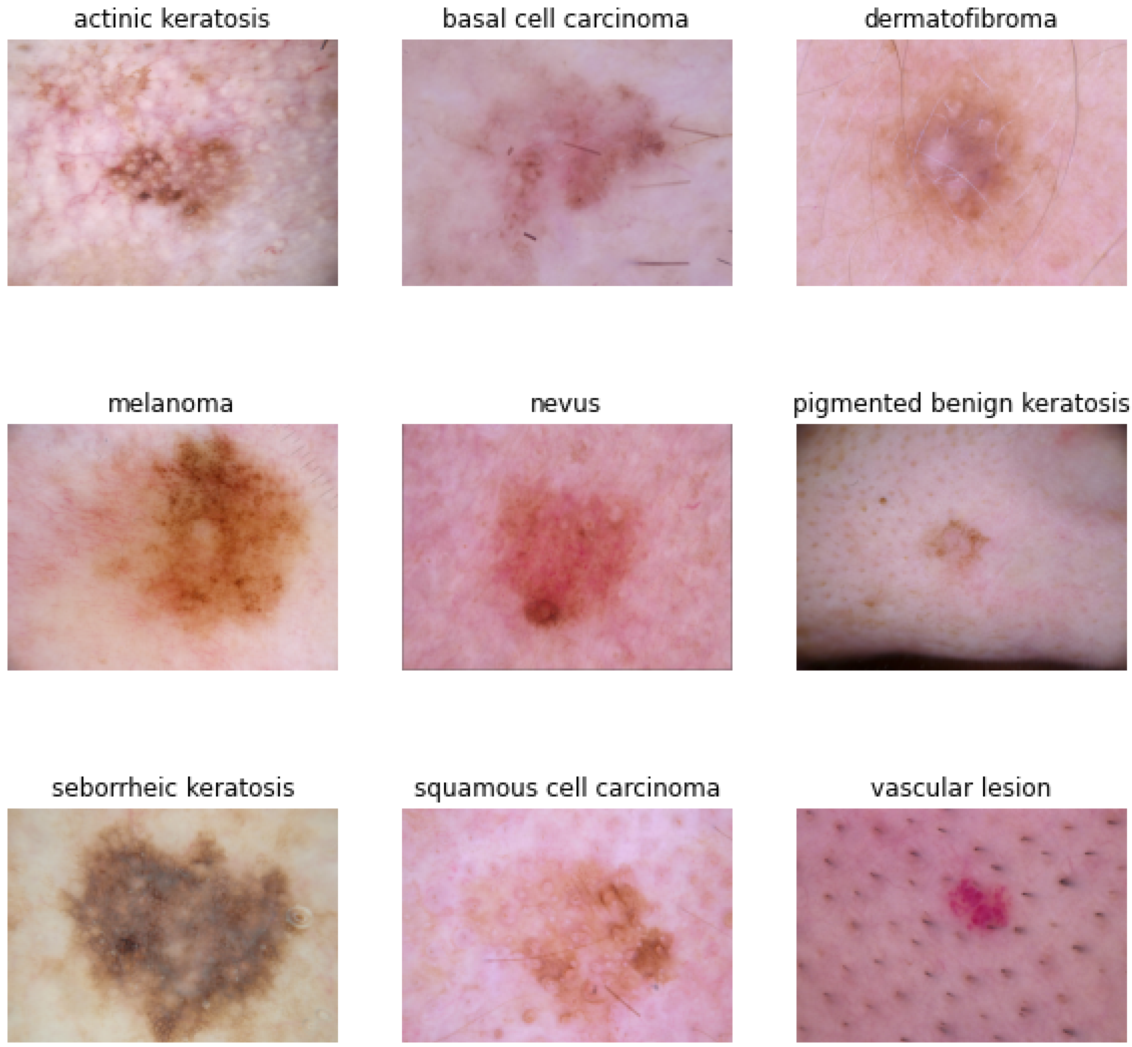
Observation : Melanoma has 20% of data in train and 0.71% data in test data set.

Highest Sample of Data : pigmented benign keratosis

Lowest Sample of Data : seborrheic keratosis

DataSet Visualization

```
#get one image from each folder
import glob
import matplotlib.image as mpimg

file_path = []
class_name = []

#get one file path from each folder
for dir_name in dir_train:
  path = str(data_dir_train) +"/"+ dir_name
  for file_name in glob.iglob(path+'/*.jpg', recursive=True):
    #print(file_name)
    file_path.append(file_name)
    class_name.append(dir_name)
    break


#display one image from each folder
plt.figure(figsize=(10,10))
for i in range(len(class_name)):
  ax = plt.subplot(3,3,i+1)
  img = mpimg.imread(file_path[i])
  plt.imshow(img)
  plt.axis("off")
  plt.title(class_name[i])
```



Load Images For the CNN Model Inputs

```
#data loader params
batch_size = 32
img_height = 180
img_width = 180
```

```
# load train dataset in batches of size 32, resize the image into 180*180 pixel
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    validation_split=0.2,
    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.
```

```
# load validation dataset in batches of size 32, resize the image into 180*180 pixel
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.
```

```
# its a multiclassifier so lets see its number of different labels / classes

num_classes = len(val_ds.class_names)
num_classes
```

```
9
```

```
#class names
val_ds.class_names
```

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

Configure Dataset for Performance

```
#Dataset.cache() keeps the images in memory after they're loaded off disk during the first
#Dataset.prefetch() overlaps data preprocessing and model execution while training.

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

M1 Model (Base Model)

```
#model design ( CNN Model)

model = Sequential([
         layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
```

```
        layers.Conv2D(32,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        layers.Conv2D(64,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        layers.Flatten(),
        layers.Dense(128,activation="relu"),
        layers.Dense(num_classes)
])


# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l


#model design summary

model.summary()
```

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

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 flatten (Flatten)           (None, 30976)             0

 dense (Dense)               (None, 128)               3965056

 dense_1 (Dense)             (None, 9)                 1161

=================================================================
Total params: 3,989,801
Trainable params: 3,989,801
Non-trainable params: 0
_________________________________________________________________
```

```
#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

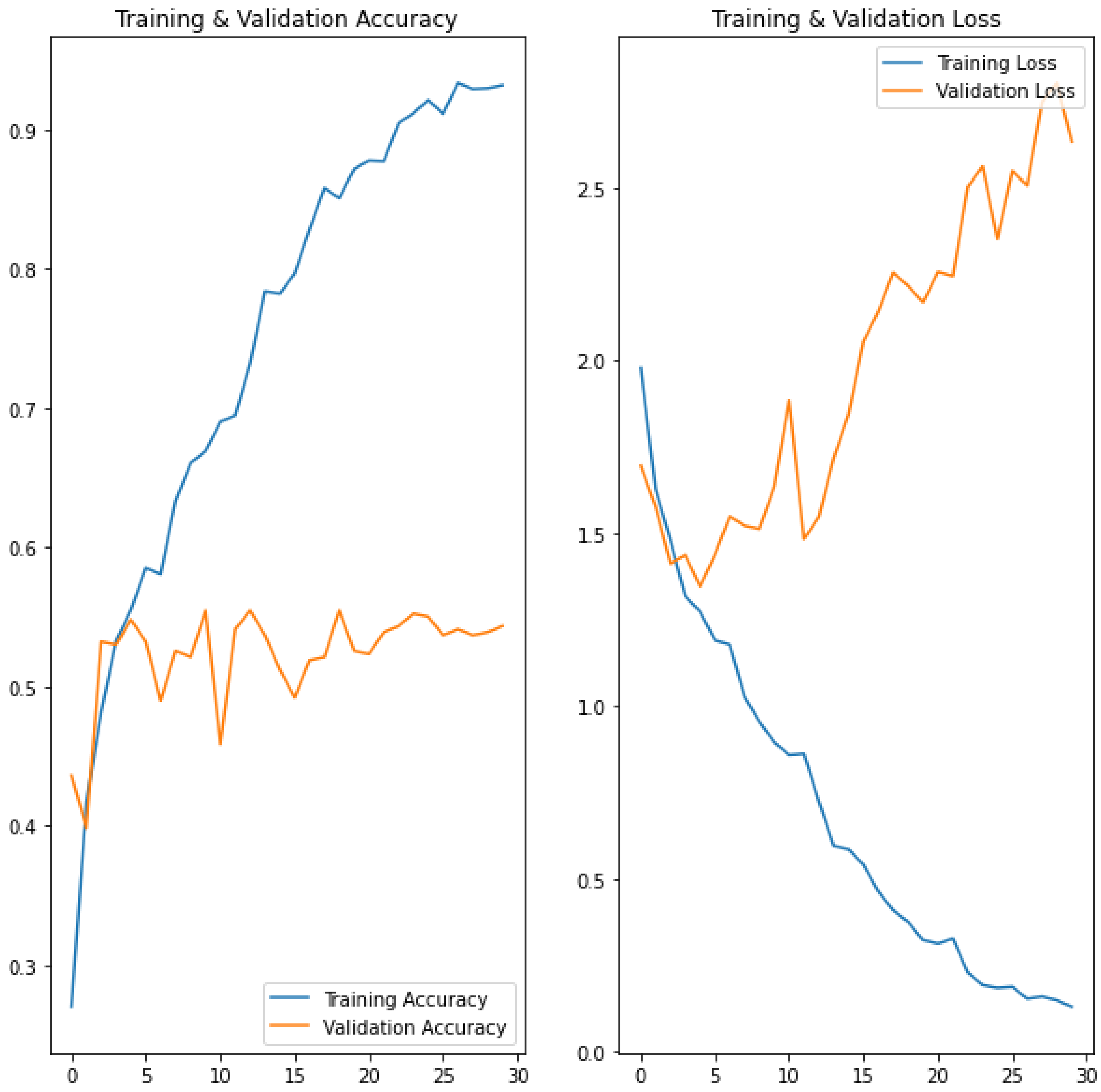
```
Epoch 2/30
56/56 [==============================] - 3s 61ms/step - loss: 1.6267 - accuracy:
Epoch 3/30
```

```
Epoch 3/30
56/56 [==============================] - 3s 61ms/step - loss: 1.4790 - accuracy:
Epoch 4/30
56/56 [==============================] - 3s 61ms/step - loss: 1.3179 - accuracy:
Epoch 5/30
56/56 [==============================] - 3s 62ms/step - loss: 1.2732 - accuracy:
Epoch 6/30
56/56 [==============================] - 3s 62ms/step - loss: 1.1905 - accuracy:
Epoch 7/30
56/56 [==============================] - 3s 61ms/step - loss: 1.1780 - accuracy:
Epoch 8/30
56/56 [==============================] - 3s 62ms/step - loss: 1.0266 - accuracy:
Epoch 9/30
56/56 [==============================] - 3s 62ms/step - loss: 0.9542 - accuracy:
Epoch 10/30
56/56 [==============================] - 3s 61ms/step - loss: 0.8953 - accuracy:
Epoch 11/30
56/56 [==============================] - 3s 61ms/step - loss: 0.8581 - accuracy:
Epoch 12/30
56/56 [==============================] - 3s 61ms/step - loss: 0.8616 - accuracy:
Epoch 13/30
56/56 [==============================] - 3s 62ms/step - loss: 0.7241 - accuracy:
Epoch 14/30
56/56 [==============================] - 3s 61ms/step - loss: 0.5951 - accuracy:
Epoch 15/30
56/56 [==============================] - 3s 61ms/step - loss: 0.5849 - accuracy:
Epoch 16/30
56/56 [==============================] - 3s 62ms/step - loss: 0.5407 - accuracy:
Epoch 17/30
56/56 [==============================] - 3s 62ms/step - loss: 0.4632 - accuracy:
Epoch 18/30
56/56 [==============================] - 3s 61ms/step - loss: 0.4089 - accuracy:
Epoch 19/30
56/56 [==============================] - 3s 61ms/step - loss: 0.3768 - accuracy:
Epoch 20/30
56/56 [==============================] - 3s 61ms/step - loss: 0.3235 - accuracy:
Epoch 21/30
56/56 [==============================] - 3s 62ms/step - loss: 0.3133 - accuracy:
Epoch 22/30
56/56 [==============================] - 3s 61ms/step - loss: 0.3280 - accuracy:
Epoch 23/30
56/56 [==============================] - 3s 61ms/step - loss: 0.2297 - accuracy:
Epoch 24/30
56/56 [==============================] - 3s 61ms/step - loss: 0.1933 - accuracy:
Epoch 25/30
56/56 [==============================] - 3s 62ms/step - loss: 0.1853 - accuracy:
Epoch 26/30
56/56 [==============================] - 3s 61ms/step - loss: 0.1885 - accuracy:
Epoch 27/30
56/56 [==============================] - 3s 62ms/step - loss: 0.1536 - accuracy:
Epoch 28/30
56/56 [==============================] - 3s 61ms/step - loss: 0.1601 - accuracy:
Epoch 29/30
56/56 [==============================] - 3s 61ms/step - loss: 0.1493 - accuracy:
Epoch 30/30
56/56 [==============================] - 3s 62ms/step - loss: 0.1303 - accuracy:
```

```
# accuracy & loss graph
```

```
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=(10,10))
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 & 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 & Validation Loss')
```

```
Text(0.5, 1.0, 'Training & Validation Loss')
```



Observation

1. Training Accuracy : Training Accuracy is high
2. Validation Accuracy : Validation accuracy is low compared to the Training Accuracy so , its not a good model.
3. Training Loss : Its decerasing
4. Validation Loss : its increasing per epoch so not a good fit

M2 Model (With Augumentation)

```
data_augument = keras.Sequential([
                             layers.experimental.preprocessing.RandomFlip(mode="horizontal
                             layers.experimental.preprocessing.RandomRotation(0.2, fill_mo
                             layers.experimental.preprocessing.RandomZoom(height_factor=(0
])
```

```
model = Sequential([
         data_augument,
         layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Conv2D(64,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.Flatten(),
         layers.Dense(128,activation="relu"),
         layers.Dense(num_classes)
])
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16)       0         
 2D)
```

```
 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 30976)             0

 dense_2 (Dense)             (None, 128)               3965056

 dense_3 (Dense)             (None, 9)                 1161

=================================================================
Total params: 3,989,801
Trainable params: 3,989,801
Non-trainable params: 0
_________________________________________________________________
```

```
# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l
```

```
#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

```
56/56 [==============================] - 4s 72ms/step - loss: 1.6642 - accuracy:
Epoch 3/30
56/56 [==============================] - 4s 72ms/step - loss: 1.4751 - accuracy:
Epoch 4/30
56/56 [==============================] - 4s 72ms/step - loss: 1.4068 - accuracy:
Epoch 5/30
56/56 [==============================] - 4s 72ms/step - loss: 1.3411 - accuracy:
Epoch 6/30
56/56 [==============================] - 4s 71ms/step - loss: 1.2912 - accuracy:
Epoch 7/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2644 - accuracy:
Epoch 8/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2432 - accuracy:
Epoch 9/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2152 - accuracy:
Epoch 10/30
56/56 [==============================] - 4s 71ms/step - loss: 1.2724 - accuracy:
Epoch 11/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2168 - accuracy:
Epoch 12/30
56/56 [==============================] - 4s 71ms/step - loss: 1.2115 - accuracy:
Epoch 13/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2340 - accuracy:
Epoch 14/30

56/56 [==============================] - 4s 71ms/step - loss: 1.1695 - accuracy:
Epoch 15/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1577 - accuracy:
Epoch 16/30
```

```
Epoch 16/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1490 - accuracy:
Epoch 17/30
56/56 [==============================] - 4s 71ms/step - loss: 1.1687 - accuracy:
Epoch 18/30
56/56 [==============================] - 4s 71ms/step - loss: 1.1437 - accuracy:
Epoch 19/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1888 - accuracy:
Epoch 20/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0977 - accuracy:
Epoch 21/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0857 - accuracy:
Epoch 22/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1049 - accuracy:
Epoch 23/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0744 - accuracy:
Epoch 24/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0702 - accuracy:
Epoch 25/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0326 - accuracy:
Epoch 26/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0464 - accuracy:
Epoch 27/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0609 - accuracy:
Epoch 28/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0310 - accuracy:
Epoch 29/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0359 - accuracy:
Epoch 30/30
```

Observation : just by adding augumentation it wont help us , so lets add the drop out as well

M3 Model (With Augumentation & droupout)

```
model = Sequential([
          data_augument,
          layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
          layers.Conv2D(16,3,padding='same',activation="relu"),
          layers.MaxPool2D((2,2),strides=2),
          layers.Conv2D(32,3,padding='same',activation="relu"),
          layers.MaxPool2D((2,2),strides=2),
          layers.Conv2D(64,3,padding='same',activation="relu"),
          layers.MaxPool2D((2,2),strides=2),
          layers.Dropout(0.2), # droupout layer
          layers.Flatten(),
          layers.Dense(128,activation="relu"),
```

```
        layers.Dense(num_classes)

model.summary()
```

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

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

 conv2d_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)

 conv2d_8 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_8 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 dropout (Dropout)           (None, 22, 22, 64)        0

 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
_________________________________________________________________
```

```
# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l
```

```
#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

```
56/56 [==============================] - 4s 72ms/step - loss: 1.8070 - accuracy:
Epoch 3/30
56/56 [==============================] - 4s 72ms/step - loss: 1.6268 - accuracy:
Epoch 4/30
56/56 [==============================] - 4s 72ms/step - loss: 1.5503 - accuracy:

Epoch 5/30
56/56 [==============================] - 4s 72ms/step - loss: 1.5193 - accuracy:
Epoch 6/30
56/56 [==============================] - 4s 72ms/step - loss: 1.4204 - accuracy:
```

```
56/56 [==============================] - 4s 72ms/step - loss: 1.4204 - accuracy:
Epoch 7/30
56/56 [==============================] - 4s 72ms/step - loss: 1.3884 - accuracy:
Epoch 8/30
56/56 [==============================] - 4s 72ms/step - loss: 1.3610 - accuracy:
Epoch 9/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2956 - accuracy:
Epoch 10/30
56/56 [==============================] - 4s 74ms/step - loss: 1.2662 - accuracy:
Epoch 11/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2630 - accuracy:
Epoch 12/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2597 - accuracy:
Epoch 13/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1909 - accuracy:
Epoch 14/30
56/56 [==============================] - 4s 73ms/step - loss: 1.1862 - accuracy:
Epoch 15/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1769 - accuracy:
Epoch 16/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2413 - accuracy:
Epoch 17/30
56/56 [==============================] - 4s 72ms/step - loss: 1.2097 - accuracy:
Epoch 18/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1679 - accuracy:
Epoch 19/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1806 - accuracy:
Epoch 20/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1545 - accuracy:
Epoch 21/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1353 - accuracy:
Epoch 22/30
56/56 [==============================] - 4s 72ms/step - loss: 1.1279 - accuracy:
Epoch 23/30
56/56 [==============================] - 4s 71ms/step - loss: 1.1216 - accuracy:
Epoch 24/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0933 - accuracy:
Epoch 25/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0578 - accuracy:
Epoch 26/30
56/56 [==============================] - 4s 73ms/step - loss: 1.0982 - accuracy:
Epoch 27/30
56/56 [==============================] - 4s 73ms/step - loss: 1.0685 - accuracy:
Epoch 28/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0825 - accuracy:
Epoch 29/30
56/56 [==============================] - 4s 72ms/step - loss: 1.0932 - accuracy:
Epoch 30/30
```

```
# slight increase in accuracy, so lets add droupout to More Layers
```

M4 Model (with Augumentation + Droupouts (to additional Layers))

```
model = Sequential([
        data_augument,
        layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
        layers.Conv2D(16,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),

        layers.Conv2D(32,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        layers.Dropout(0.25), # droupout layer

        layers.Conv2D(64,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        layers.Dropout(0.25), # droupout layer

        layers.Conv2D(128,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        layers.Dropout(0.25), # droupout layer

        layers.Flatten(),
        layers.Dense(128,activation="relu"),
        layers.Dropout(0.25), # droupout layer

        layers.Dense(num_classes)
])


# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l


#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

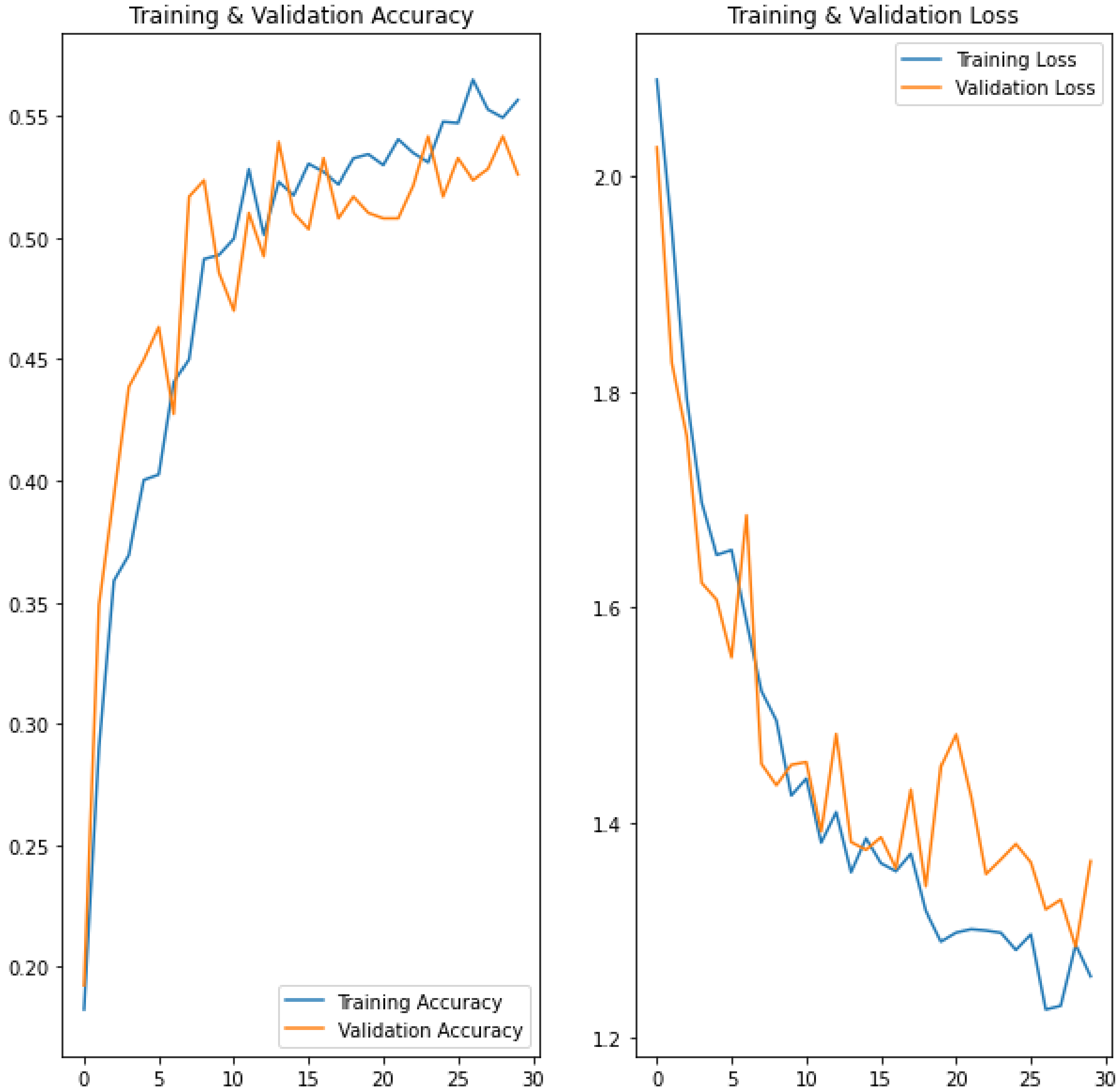
```
56/56 [==============================] - 4s 78ms/step - loss: 1.9514 - accuracy:
Epoch 3/30
56/56 [==============================] - 4s 78ms/step - loss: 1.7939 - accuracy:
Epoch 4/30
56/56 [==============================] - 4s 78ms/step - loss: 1.6970 - accuracy:
Epoch 5/30
56/56 [==============================] - 4s 78ms/step - loss: 1.6485 - accuracy:
Epoch 6/30
56/56 [==============================] - 4s 78ms/step - loss: 1.6531 - accuracy:
Epoch 7/30
56/56 [==============================] - 4s 78ms/step - loss: 1.5867 - accuracy:
Epoch 8/30
56/56 [==============================] - 4s 78ms/step - loss: 1.5219 - accuracy:
Epoch 9/30
56/56 [==============================] - 4s 79ms/step - loss: 1.4945 - accuracy:
Epoch 10/30
56/56 [==============================] - 4s 79ms/step - loss: 1.4247 - accuracy:
Epoch 11/30
56/56 [==============================] - 4s 79ms/step - loss: 1.4405 - accuracy:
Epoch 12/30
56/56 [==============================] - 4s 78ms/step - loss: 1.3808 - accuracy:
Epoch 13/30
56/56 [==============================] - 4s 79ms/step - loss: 1.4093 - accuracy:
```

```
56/56 [==============================] - 4s 79ms/step - loss: 1.4093 - accuracy:
Epoch 14/30
56/56 [==============================] - 4s 78ms/step - loss: 1.3534 - accuracy:
Epoch 15/30
56/56 [==============================] - 4s 79ms/step - loss: 1.3849 - accuracy:
Epoch 16/30
56/56 [==============================] - 4s 79ms/step - loss: 1.3618 - accuracy:
Epoch 17/30
56/56 [==============================] - 4s 79ms/step - loss: 1.3543 - accuracy:
Epoch 18/30
56/56 [==============================] - 4s 78ms/step - loss: 1.3706 - accuracy:
Epoch 19/30
56/56 [==============================] - 4s 78ms/step - loss: 1.3176 - accuracy:
Epoch 20/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2893 - accuracy:
Epoch 21/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2977 - accuracy:
Epoch 22/30
56/56 [==============================] - 4s 78ms/step - loss: 1.3008 - accuracy:
Epoch 23/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2996 - accuracy:
Epoch 24/30
56/56 [==============================] - 4s 79ms/step - loss: 1.2975 - accuracy:
Epoch 25/30
56/56 [==============================] - 4s 79ms/step - loss: 1.2815 - accuracy:
Epoch 26/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2959 - accuracy:
Epoch 27/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2261 - accuracy:
Epoch 28/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2295 - accuracy:
Epoch 29/30
56/56 [==============================] - 4s 79ms/step - loss: 1.2864 - accuracy:
Epoch 30/30
56/56 [==============================] - 4s 78ms/step - loss: 1.2569 - accuracy:
```

```
# accuracy & loss graph

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=(10,10))
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 & 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 & Validation Loss')
```

```
Text(0.5, 1.0, 'Training & Validation Loss')
```



Observation : Now model has no Overfitting : as both train & validation accuracy overlap

M5 model : Additional Experiment with Dropouts

```
model = Sequential([
        data_augument,
        layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
        layers.Conv2D(16,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),

        layers.Conv2D(32,3,padding='same',activation="relu"),
```

```
        layers.MaxPool2D((2,2),strides=2),
        #layers.Dropout(0.25), # droupout layer

        layers.Conv2D(64,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        #layers.Dropout(0.25), # droupout layer

        layers.Conv2D(128,3,padding='same',activation="relu"),
        layers.MaxPool2D((2,2),strides=2),
        #layers.Dropout(0.25), # droupout layer

        layers.Flatten(),
        layers.Dense(128,activation="relu"),
        layers.Dropout(0.25), # droupout layer

        layers.Dense(num_classes)
])


#model design overview

model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_4 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_13 (Conv2D)          (None, 180, 180, 16)      448

 max_pooling2d_13 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 90, 90, 32)        4640

 max_pooling2d_14 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 conv2d_15 (Conv2D)          (None, 45, 45, 64)        18496

 max_pooling2d_15 (MaxPoolin  (None, 22, 22, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 22, 22, 128)       73856

 max_pooling2d_16 (MaxPoolin  (None, 11, 11, 128)      0
 g2D)

 flatten_4 (Flatten)         (None, 15488)             0

 dense_8 (Dense)             (None, 128)               1982592

 dropout_5 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 9)                 1161
```

```
=================================================================
Total params: 2,081,193
Trainable params: 2,081,193
Non-trainable params: 0
_________________________________________________________________
```

```
# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l
```

```
#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

```
56/56 [==============================] - 4s 76ms/step - loss: 1.9310 - accuracy:
Epoch 3/30
56/56 [==============================] - 4s 76ms/step - loss: 1.8487 - accuracy:
Epoch 4/30
56/56 [==============================] - 4s 76ms/step - loss: 1.7490 - accuracy:
Epoch 5/30
56/56 [==============================] - 4s 76ms/step - loss: 1.7095 - accuracy:
Epoch 6/30
56/56 [==============================] - 4s 76ms/step - loss: 1.6117 - accuracy:
Epoch 7/30
56/56 [==============================] - 4s 76ms/step - loss: 1.5640 - accuracy:
Epoch 8/30
56/56 [==============================] - 4s 76ms/step - loss: 1.4807 - accuracy:
Epoch 9/30
56/56 [==============================] - 4s 76ms/step - loss: 1.5099 - accuracy:
Epoch 10/30
56/56 [==============================] - 4s 76ms/step - loss: 1.4322 - accuracy:
Epoch 11/30
56/56 [==============================] - 5s 82ms/step - loss: 1.3860 - accuracy:
Epoch 12/30
56/56 [==============================] - 4s 80ms/step - loss: 1.3742 - accuracy:
Epoch 13/30
56/56 [==============================] - 4s 77ms/step - loss: 1.3199 - accuracy:
Epoch 14/30
56/56 [==============================] - 4s 76ms/step - loss: 1.3531 - accuracy:
Epoch 15/30
56/56 [==============================] - 4s 76ms/step - loss: 1.3109 - accuracy:
Epoch 16/30
56/56 [==============================] - 4s 76ms/step - loss: 1.3093 - accuracy:
Epoch 17/30
56/56 [==============================] - 4s 76ms/step - loss: 1.3102 - accuracy:
Epoch 18/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2752 - accuracy:
Epoch 19/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2623 - accuracy:
Epoch 20/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2813 - accuracy:
Epoch 21/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2491 - accuracy:

Epoch 22/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2326 - accuracy:
Epoch 23/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2307 - accuracy:
```

```
Epoch 24/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2089 - accuracy:
Epoch 25/30
56/56 [==============================] - 4s 76ms/step - loss: 1.1918 - accuracy:
Epoch 26/30
56/56 [==============================] - 4s 76ms/step - loss: 1.2297 - accuracy:
Epoch 27/30
56/56 [==============================] - 4s 75ms/step - loss: 1.2222 - accuracy:
Epoch 28/30
56/56 [==============================] - 4s 76ms/step - loss: 1.1955 - accuracy:
Epoch 29/30
56/56 [==============================] - 4s 76ms/step - loss: 1.1615 - accuracy:
Epoch 30/30
```

M6 Model (Augumetation + Batch Normalization + Droupouts)

```
model = Sequential([
         data_augument,
         layers.Rescaling(1./255,input_shape=(img_height,img_width,3)),
         layers.Conv2D(16,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(32,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(64,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Conv2D(128,3,padding='same',activation="relu"),
         layers.MaxPool2D((2,2),strides=2),
         layers.BatchNormalization(),
         layers.Dropout(0.25), # droupout layer

         layers.Flatten(),
         layers.Dense(128,activation="relu"),

         layers.Dense(num_classes)
])
```

```
model.summary()
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_5 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_17 (Conv2D)          (None, 180, 180, 16)      448

 max_pooling2d_17 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

 batch_normalization (BatchN  (None, 90, 90, 16)       64
 ormalization)

 dropout_6 (Dropout)         (None, 90, 90, 16)        0

 conv2d_18 (Conv2D)          (None, 90, 90, 32)        4640

 max_pooling2d_18 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 batch_normalization_1 (Batc  (None, 45, 45, 32)       128
 hNormalization)

 dropout_7 (Dropout)         (None, 45, 45, 32)        0

 conv2d_19 (Conv2D)          (None, 45, 45, 64)        18496

 max_pooling2d_19 (MaxPoolin  (None, 22, 22, 64)       0
 g2D)

 batch_normalization_2 (Batc  (None, 22, 22, 64)       256
 hNormalization)

 dropout_8 (Dropout)         (None, 22, 22, 64)        0

 conv2d_20 (Conv2D)          (None, 22, 22, 128)       73856

 max_pooling2d_20 (MaxPoolin  (None, 11, 11, 128)      0
 g2D)

 batch_normalization_3 (Batc  (None, 11, 11, 128)      512
 hNormalization)

 dropout_9 (Dropout)         (None, 11, 11, 128)       0

 flatten_5 (Flatten)         (None, 15488)             0

 dense_10 (Dense)            (None, 128)               1982592

 dense_11 (Dense)            (None, 9)                 1161

=================================================================
Total params: 2,082,153
Trainable params: 2,081,673
Non-trainable params: 480
```

```
# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l


#train the model : run the model on train & validation set
epochs = 30
history = model.fit( train_ds , validation_data= val_ds , epochs = epochs)
```

```
56/56 [==============================] - 5s 83ms/step - loss: 1.6514 - accuracy:
Epoch 3/30
56/56 [==============================] - 5s 83ms/step - loss: 1.5596 - accuracy:
Epoch 4/30
56/56 [==============================] - 5s 84ms/step - loss: 1.4726 - accuracy:
Epoch 5/30
56/56 [==============================] - 5s 83ms/step - loss: 1.4198 - accuracy:
Epoch 6/30
56/56 [==============================] - 5s 86ms/step - loss: 1.3324 - accuracy:
Epoch 7/30
56/56 [==============================] - 5s 85ms/step - loss: 1.3060 - accuracy:
Epoch 8/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2977 - accuracy:
Epoch 9/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2520 - accuracy:
Epoch 10/30
56/56 [==============================] - 5s 84ms/step - loss: 1.2858 - accuracy:
Epoch 11/30
56/56 [==============================] - 5s 84ms/step - loss: 1.2536 - accuracy:
Epoch 12/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2056 - accuracy:
Epoch 13/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2390 - accuracy:
Epoch 14/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2030 - accuracy:
Epoch 15/30
56/56 [==============================] - 5s 83ms/step - loss: 1.2054 - accuracy:
Epoch 16/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1974 - accuracy:
Epoch 17/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1658 - accuracy:
Epoch 18/30
56/56 [==============================] - 5s 84ms/step - loss: 1.1737 - accuracy:
Epoch 19/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1488 - accuracy:
Epoch 20/30
56/56 [==============================] - 5s 84ms/step - loss: 1.1697 - accuracy:
Epoch 21/30
56/56 [==============================] - 5s 84ms/step - loss: 1.1293 - accuracy:
Epoch 22/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1543 - accuracy:
Epoch 23/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1068 - accuracy:
Epoch 24/30
56/56 [==============================] - 5s 83ms/step - loss: 1.1094 - accuracy:
Epoch 25/30
56/56 [==============================] - 5s 83ms/step - loss: 1.0877 - accuracy:

Epoch 26/30
56/56 [==============================] - 5s 94ms/step - loss: 1.0667 - accuracy:
Epoch 27/30
56/56 [==============================] - 5s 86ms/step - loss: 1.0944 - accuracy:
```

```
56/56 [==============================] - 5s 86ms/step - loss: 1.0944 - accuracy:
Epoch 28/30
56/56 [==============================] - 5s 84ms/step - loss: 1.0733 - accuracy:
Epoch 29/30
56/56 [==============================] - 5s 83ms/step - loss: 1.0406 - accuracy:
Epoch 30/30
56/56 [==============================] - 5s 83ms/step - loss: 1.0789 - accuracy:
```

```
# Observation : No Additional improvement, its due to very less data points so lets increa
```

Using Another Way of Augmentation to Handle Class Imbalance

Using Augmentor Pipeline (Add additional Images)

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

```
Collecting Augmentor
  Downloading Augmentor-0.2.9-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: future>=0.16.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.7/dist-package
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.9
```

```
import Augmentor
```

```
# add 500 new sample to each folder
for class_name in data_detail_pd.index:
  #print(class_name)
  p = Augmentor.Pipeline(str(data_dir_train)+"/"+class_name,save_format='.jpg')
  p.rotate(probability=0.7,max_left_rotation=10,max_right_rotation=10)
  p.sample(500)
```

```
Initialised with 114 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 376 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 95 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 438 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 357 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 462 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
```

```
Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 181 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
Initialised with 139 image(s) found.
Output directory set to /content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cance
```

```
data_detail_pd.index
```

```
Index(['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma',
       'melanoma', 'nevus', 'pigmented benign keratosis',
       'seborrheic keratosis', 'squamous cell carcinoma', 'vascular lesion'],
      dtype='object', name='Dir_Name')
```

```
data_dir_train
```

```
PosixPath('/content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cancer ISIC The Ir
```

```
#count of additional images added

additional_images_added = len(list(data_dir_train.glob("*/output/*jpg")))
additional_images_added
```

```
4500
```

Now train the Model on the Additional Images Obtained via Augmentor (4500 Images) + Original Images (2239 Images)

```
# we need to reinitalize the train_ds & val_ds
train_ds_new = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    validation_split=0.2,
    subset = "training",
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

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

```
#validation dataset

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

```
# AutoTune & cache for performance

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

```
# Model Defination

model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.BatchNormalization(),
  layers.Dropout(0.25),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.BatchNormalization(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.25),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

```
# model compilation

model.compile(optimizer="adam",loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l
```

```
#model design

model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_6 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_21 (Conv2D)          (None, 180, 180, 16)      448
```

```
 max_pooling2d_21 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

 batch_normalization_4 (Batc  (None, 90, 90, 16)       64
 hNormalization)

 dropout_10 (Dropout)        (None, 90, 90, 16)        0

 conv2d_22 (Conv2D)          (None, 90, 90, 32)        4640

 max_pooling2d_22 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 batch_normalization_5 (Batc  (None, 45, 45, 32)       128
 hNormalization)

 conv2d_23 (Conv2D)          (None, 45, 45, 64)        18496

 max_pooling2d_23 (MaxPoolin  (None, 22, 22, 64)       0
 g2D)

 dropout_11 (Dropout)        (None, 22, 22, 64)        0

 flatten_6 (Flatten)         (None, 30976)             0

 dense_12 (Dense)            (None, 128)               3965056

 dense_13 (Dense)            (None, 9)                 1161

=================================================================
Total params: 3,989,993
Trainable params: 3,989,897
Non-trainable params: 96
_________________________________________________________________
```

```
# run the model to fit train datapoint and check accuracy on validation dataset

epochs = 30
history = model.fit(
  train_ds_new,
  validation_data=val_ds_new,
  epochs=epochs
)
```

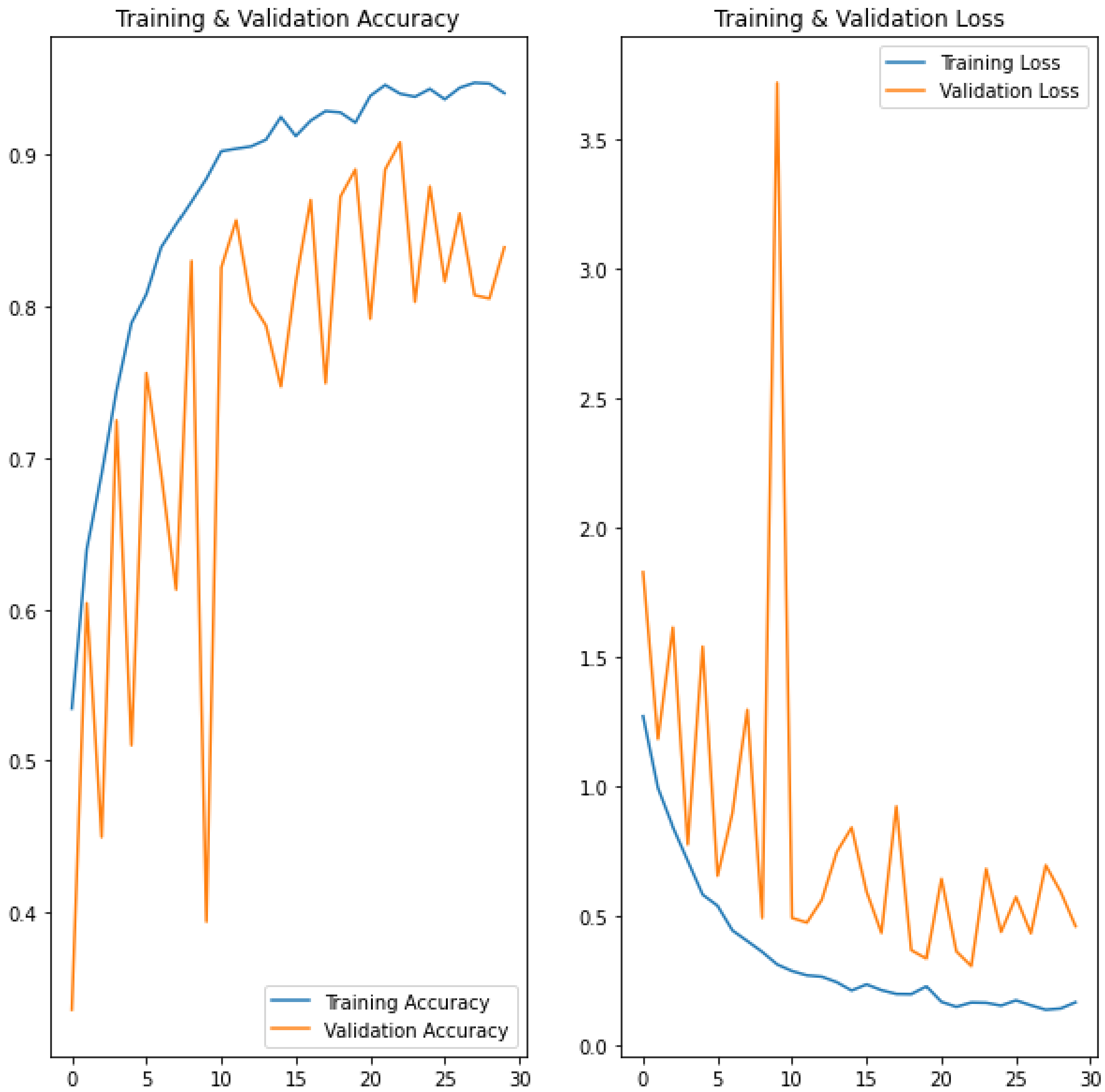
```
169/169 [==============================] - 34s 191ms/step - loss: 0.9921 - accura
Epoch 3/30
169/169 [==============================] - 34s 192ms/step - loss: 0.8423 - accura
Epoch 4/30
169/169 [==============================] - 35s 197ms/step - loss: 0.7116 - accura
Epoch 5/30
169/169 [==============================] - 34s 194ms/step - loss: 0.5812 - accura
Epoch 6/30
169/169 [==============================] - 36s 205ms/step - loss: 0.5382 - accura
Epoch 7/30

169/169 [==============================] - 35s 201ms/step - loss: 0.4427 - accura
Epoch 8/30
169/169 [==============================] - 36s 204ms/step - loss: 0.4018 - accura
Epoch 9/30
```

```
Epoch 9/30
169/169 [==============================] - 37s 211ms/step - loss: 0.3598 - accura
Epoch 10/30
169/169 [==============================] - 36s 205ms/step - loss: 0.3114 - accura
Epoch 11/30
169/169 [==============================] - 37s 210ms/step - loss: 0.2858 - accura
Epoch 12/30
169/169 [==============================] - 38s 214ms/step - loss: 0.2691 - accura
Epoch 13/30
169/169 [==============================] - 35s 196ms/step - loss: 0.2639 - accura
Epoch 14/30
169/169 [==============================] - 37s 209ms/step - loss: 0.2428 - accura
Epoch 15/30
169/169 [==============================] - 37s 210ms/step - loss: 0.2104 - accura
Epoch 16/30
169/169 [==============================] - 35s 197ms/step - loss: 0.2342 - accura
Epoch 17/30
169/169 [==============================] - 35s 201ms/step - loss: 0.2119 - accura
Epoch 18/30
169/169 [==============================] - 35s 198ms/step - loss: 0.1972 - accura
Epoch 19/30
169/169 [==============================] - 35s 202ms/step - loss: 0.1963 - accura
Epoch 20/30
169/169 [==============================] - 35s 200ms/step - loss: 0.2260 - accura
Epoch 21/30
169/169 [==============================] - 34s 195ms/step - loss: 0.1666 - accura
Epoch 22/30
169/169 [==============================] - 34s 195ms/step - loss: 0.1473 - accura
Epoch 23/30
169/169 [==============================] - 35s 200ms/step - loss: 0.1641 - accura
Epoch 24/30
169/169 [==============================] - 36s 206ms/step - loss: 0.1629 - accura
Epoch 25/30
169/169 [==============================] - 34s 194ms/step - loss: 0.1521 - accura
Epoch 26/30
169/169 [==============================] - 34s 196ms/step - loss: 0.1725 - accura
Epoch 27/30
169/169 [==============================] - 34s 197ms/step - loss: 0.1533 - accura
Epoch 28/30
169/169 [==============================] - 34s 196ms/step - loss: 0.1362 - accura
Epoch 29/30
169/169 [==============================] - 36s 204ms/step - loss: 0.1410 - accura
Epoch 30/30
169/169 [==============================] - 35s 202ms/step - loss: 0.1649 - accura
```

```
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=(10,10))
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 & 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 & Validation Loss')
```

```
Text(0.5, 1.0, 'Training & Validation Loss')
```



Now we have good train accuracy (94%) and Validation Accuracy (84%)

Analysis on Test Data

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_test,
    seed = 123,
    image_size = (img_height,img_width),
    batch_size = batch_size
)
```

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

```
loss , accuracy = model.evaluate(test_ds)
```

```
4/4 [==============================] - 23s 93ms/step - loss: 7.3863 - accuracy: 0.372
```

```
print("Accuracy on test data ", accuracy)
```

```
Accuracy on test data  0.37288135290145874
```

Prediction on New Test Data

```
melanoma_path = "/content/gdrive/MyDrive/CNN_DATASET/CNN_assignment/Skin cancer ISIC The I

img = tf.keras.utils.load_img(
    melanoma_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(score)
```

```
tf.Tensor(
[2.5021658e-16 1.0945042e-17 6.4754052e-10 2.1972028e-05 9.9483782e-01
 5.1402496e-03 6.1614842e-09 1.5912991e-23 2.4429672e-13], shape=(9,), dtype=float32)
```

```
print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(test_ds.class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to nevus with a 99.48 percent confidence.
```

✓ 0s completed at 8:06 AM