Problem statement:

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

Importing Skin Cancer Data

Importing all the important libraries

```
# https://stackoverflow.com/questions/71000120/colab-0-unimplemented-dnn-library-i
# # Check libcudnn8 version
# !apt-cache policy libcudnn8

# # Install latest version
# !apt install --allow-change-held-packages libcudnn8=8.4.1.50-1+cuda11.6

# # Export env variables
# !export PATH=/usr/local/cuda-11.4/bin${PATH:+:${PATH}}
# !export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
# !export LD_LIBRARY_PATH=/usr/local/cuda-11.4/include:$LD_LIBRARY_PATH
# !export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64

# # Install tensorflow
# !pip install tflite-model-maker==0.4.0
# !pip uninstall -y tensorflow && pip install -q tensorflow==2.9.1
# !pip install pycocotools==2.0.4
# !pip install opencv-python-headless==4.6.0.66
```

```
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /us
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/loc
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/p
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/py
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/d
Requirement already satisfied: importlib-resources in /usr/local/lib/python
Requirement already satisfied: toml in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: promise in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: dill in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: etils[epath] in /usr/local/lib/python3.7/dis
Requirement already satisfied: tensorflow-metadata in /usr/local/lib/python
Requirement already satisfied: dm-tree~=0.1.1 in /usr/local/lib/python3.7/d
Collecting packaging>=20.0
  Downloading packaging-20.9-py2.py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB 6.4 MB/s
Collecting tensorflowjs>=2.4.0
  Downloading tensorflowjs-3.18.0-py3-none-any.whl (77 kB)
     |████████████████████████████████| 77 kB 7.0 MB/s
Collecting pybind11>=2.6.0
  Downloading pybind11-2.10.0-py3-none-any.whl (213 kB)
     |████████████████████████████████| 213 kB 78.1 MB/s
Collecting protobuf<4.0.0dev,>=3.12.0
  Downloading protobuf-3.19.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux201
     |████████████████████████████████| 1.1 MB 69.0 MB/s
Collecting sounddevice>=0.4.4
  Downloading sounddevice-0.4.4-py3-none-any.whl (31 kB)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python
Building wheels for collected packages: fire, py-cpuinfo
  Building wheel for fire (setup.py) ... done
  Created wheel for fire: filename=fire-0.4.0-py2.py3-none-any.whl size=115
  Stored in directory: /root/.cache/pip/wheels/8a/67/fb/2e8a12fa16661b9d5af
  Building wheel for py-cpuinfo (setup.py) ... done
  Created wheel for py-cpuinfo: filename=py_cpuinfo-8.0.0-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/d2/f1/1f/041add21dc9c4220157
Successfully built fire py-cpuinfo
Installing collected packages: protobuf, packaging, llvmlite, numba, tf-sli
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.17.3
    Uninstalling protobuf-3.17.3:
      Successfully uninstalled protobuf-3.17.3
  Attempting uninstall: packaging
    Found existing installation: packaging 21.3
    Uninstalling packaging-21.3:
      Successfully uninstalled packaging-21.3
  Attempting uninstall: llvmlite
    Found existing installation: llvmlite 0.39.0
```

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

```
## If you are using the data by mounting the google drive, use the following :
from google.colab import drive
drive.mount('/content/gdrive')

##Ref:https://towardsdatascience.com/downloading-datasets-into-google-drive-via-go
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, cal
```

This assignment uses a dataset of about 2357 images of skin cancer types. The dataset contains 9 sub-directories in each train and test subdirectories. The 9 sub-directories contains the images of 9 skin cancer types respectively.

```
# Defining the path for train and test images
## Todo: Update the paths of the train and test dataset
data_dir_train = pathlib.Path("/content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJ
data_dir_test = pathlib.Path("/content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJM
```

```
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
```

Some analysis about no. classes

```
data_detail_pd = pd.DataFrame(columns=["Dir_Name","Total Image(Train)","Total Perc
# train data in each folders
for dir_name in glob.glob(os.path.join(data_dir_train, "*")):
```

```
    total_image_in_folder = len(glob.glob(os.path.join(dir_name, "*.jpg")))
    df = {"Dir_Name":os.path.basename(dir_name),"Total Image(Train)":total_image_in_
    data_detail_pd = data_detail_pd.append(df,ignore_index=True)
  data_detail_pd = data_detail_pd.set_index("Dir_Name")
  # test data in each folders
  for dir_name in glob.glob(os.path.join(data_dir_test, "*")):
    total_image_in_folder = len(glob.glob(os.path.join(dir_name, "*.jpg")))
    data_detail_pd.loc[os.path.basename(dir_name),"Total Image(Test)"]  = total_imag
    data_detail_pd.loc[os.path.basename(dir_name),"Total Percentage(Test)"]  = round
  # data_detail_pd = data_detail_pd.set_index("Dir_Name")
  display(data_detail_pd.sort_values(by="Total Percentage(Train)",ascending=False))
```

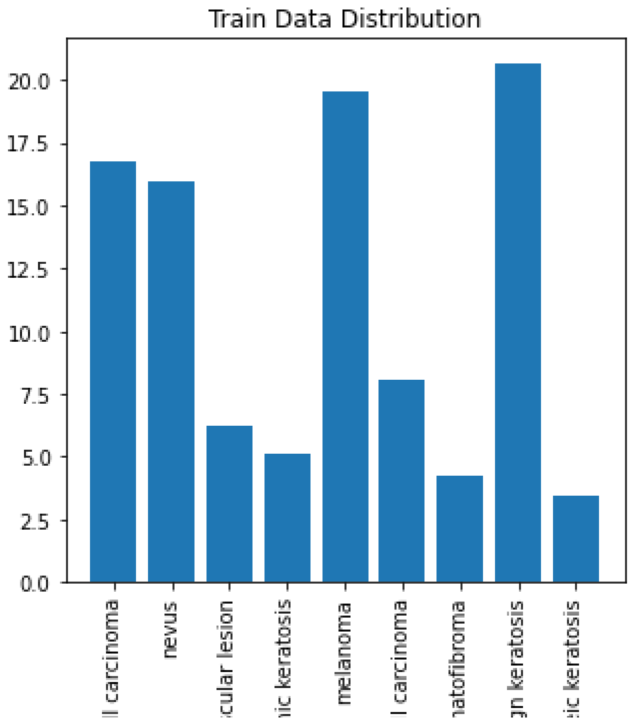
Dir_Name	Total Image(Train)	Total Percentage(Train)	Total Image(Test)	Tota Percentage(Test
pigmented benign keratosis	462	20.63	16.0	13.5
melanoma	438	19.56	16.0	13.5
basal cell carcinoma	376	16.79	16.0	13.5
nevus	357	15.94	16.0	13.5
squamous cell carcinoma	181	8.08	16.0	13.5
vascular lesion	139	6.21	3.0	2.5
actinic keratosis	114	5.09	16.0	13.5
dermatofibroma	95	4.24	16.0	13.5
seborrheic				

Observation : Melanoma has 19.56% of data in train and 13.56% data in test data set.

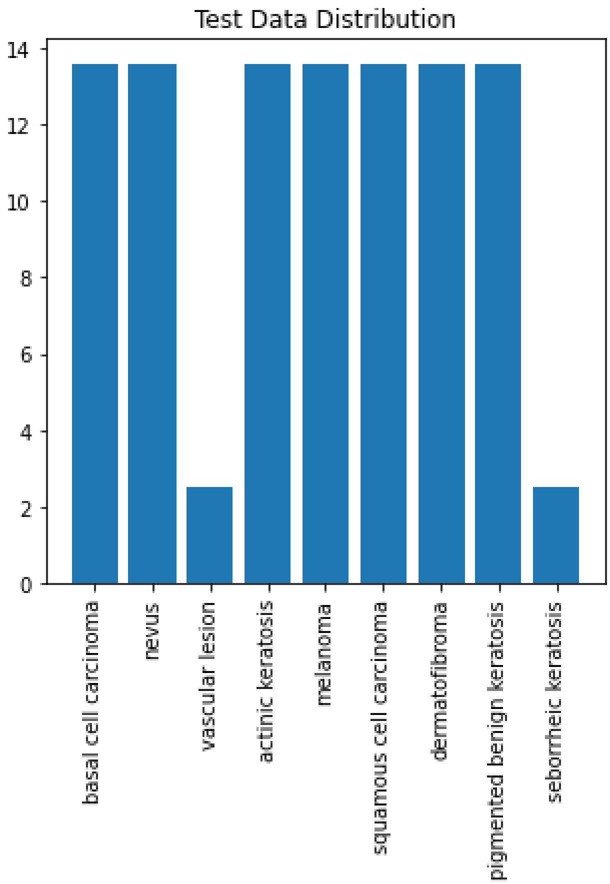
Highest Sample of Data : pigmented benign keratosis (20.63%)

Lowest Sample of Data : seborrheic keratosis (3.44% in train and 2.54% in test)

```
plt.figure(figsize=(5,5))
plt.bar(data_detail_pd.index,data_detail_pd['Total Percentage(Train)'])
plt.xticks(rotation=90)
plt.title("Train Data Distribution")
plt.show()
```




```
plt.figure(figsize=(5,5))
plt.bar(data_detail_pd.index,data_detail_pd['Total Percentage(Test)'])
plt.xticks(rotation=90)
plt.title("Test Data Distribution")
plt.show()
```



Class imbalanced, distribution of data different in train and test

Create a dataset

Define some parameters for the loader:

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

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

```
## Write your train dataset here
## Note use seed=123 while creating your dataset using tf.keras.preprocessing.imag
## Note, make sure your resize your images to the size img_height*img_width, while
train_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_train,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(img_height, img_width),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="training",
    interpolation='bicubic',
)
```

```
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.imag
## Note, make sure your resize your images to the size img_height*img_width, while
val_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_train,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(img_height, img_width),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="validation",
    interpolation='bicubic',
)
```

```
Found 2239 files belonging to 9 classes.
```

```
Using 447 files for validation.
```

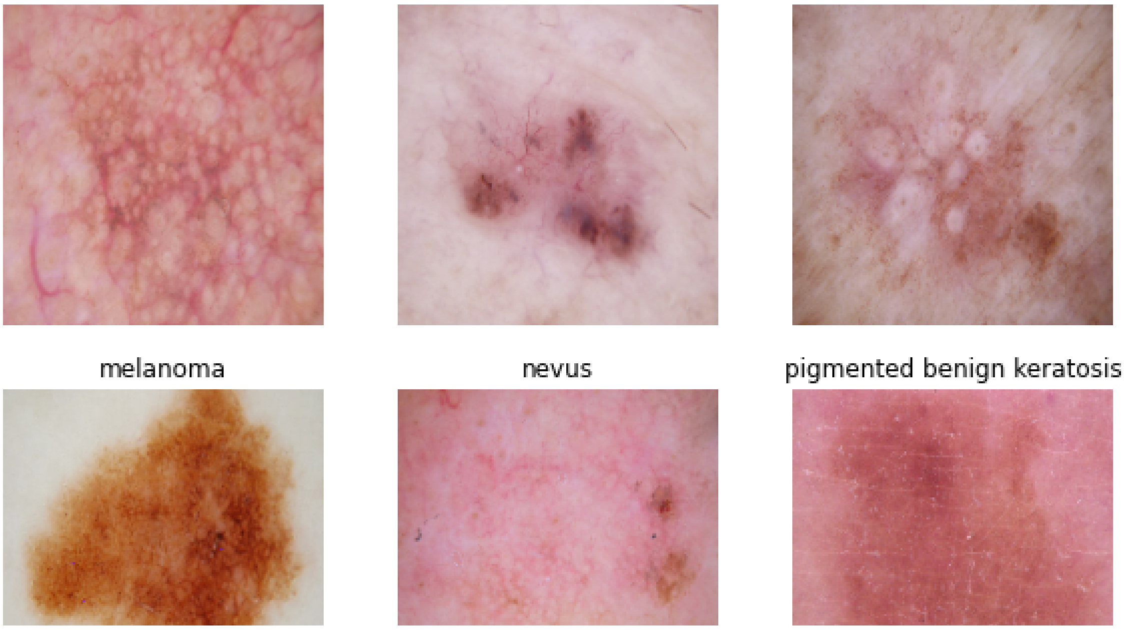
```
# List out all the classes of skin cancer and store them in a list.
# You can find the class names in the class_names attribute on these datasets.
# These correspond to the directory names in alphabetical order.
class_names = train_ds.class_names
print(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', '
```

Visualize the data

```
import matplotlib.pyplot as plt
import copy
### your code goes here, you can use training or validation data to visualize
plt.figure(figsize=(10, 10))

# get image each class
class_names_draw = [[] for i in range(len(class_names))]
count = 0
while count < len(class_names):
    for images, labels in train_ds.take(10):
        for i in range(batch_size):
            class_index = class_names.index(class_names[labels[i]])
            if class_names_draw[class_index] == []:
                class_names_draw[class_index] = images[i].numpy().astype("uint8")
                count +=1


for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(class_names_draw[i])
    plt.title(class_names[i])
    plt.axis("off")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: DeprecationW
  del sys.path[0]
```



The `image_batch` is a tensor of the shape `(32, 180, 180, 3)`. This is a batch of 32 images of shape `180x180x3` (the last dimension refers to color channels RGB). The `label_batch` is a tensor of the shape `(32,)`, these are corresponding labels to the 32 images.

`Dataset.cache()` keeps the images in memory after they're loaded off disk during the first epoch.

`Dataset.prefetch()` overlaps data preprocessing and model execution while training.

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

Create the model

```
input_shape = (180, 180, 3)
num_classes = 9
```

Model 1: Only conv2d

```
## Your code goes here
from tensorflow.keras.layers import Input, Conv2D
from tensorflow.keras.layers import MaxPool2D, Flatten, Dense, BatchNormalization,
from tensorflow.keras.layers.experimental.preprocessing import Rescaling
from tensorflow.keras import Model
```

```
model1 = Sequential([
        layers.Rescaling(1./255,input_shape=input_shape),
        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)
        ],
        name = 'Model_01')
```

Compile the model

Choose an appropirate optimiser and loss function for model training

```
### Todo, choose an appropirate optimiser and loss function
model1.compile(optimizer="adam",
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
              metrics=['accuracy'])
```

```
# View the summary of all layers
model1.summary()
```

```
Model: "Model_01"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 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
_________________________________________________________________
```

Train the model

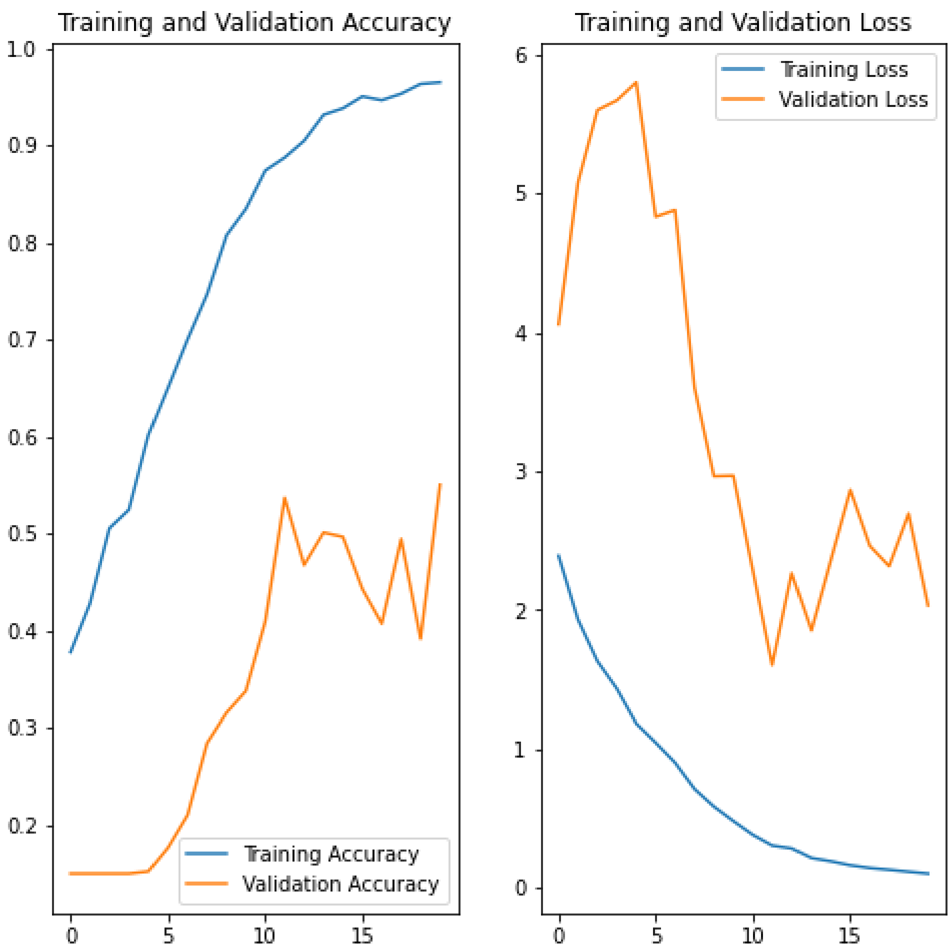
```
epochs = 20
history = model1.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

```
Epoch 1/20
56/56 [==============================] - 2s 26ms/step - loss: 2.0326 - accura
Epoch 2/20
56/56 [==============================] - 1s 23ms/step - loss: 1.7371 - accura
Epoch 3/20
56/56 [==============================] - 1s 23ms/step - loss: 1.5001 - accura
Epoch 4/20
56/56 [==============================] - 1s 23ms/step - loss: 1.3372 - accura
Epoch 5/20
56/56 [==============================] - 1s 23ms/step - loss: 1.2894 - accura
Epoch 6/20
56/56 [==============================] - 1s 23ms/step - loss: 1.1954 - accura
Epoch 7/20
56/56 [==============================] - 1s 23ms/step - loss: 1.0963 - accura
Epoch 8/20
56/56 [==============================] - 1s 23ms/step - loss: 1.0670 - accura
Epoch 9/20
56/56 [==============================] - 1s 23ms/step - loss: 1.0007 - accura
Epoch 10/20
56/56 [==============================] - 1s 23ms/step - loss: 0.8980 - accura
Epoch 11/20
56/56 [==============================] - 1s 23ms/step - loss: 0.8358 - accura
Epoch 12/20
56/56 [==============================] - 1s 23ms/step - loss: 0.7293 - accura
Epoch 13/20
56/56 [==============================] - 1s 23ms/step - loss: 0.6733 - accura
Epoch 14/20
56/56 [==============================] - 1s 23ms/step - loss: 0.5866 - accura
Epoch 15/20
56/56 [==============================] - 1s 23ms/step - loss: 0.5763 - accura
Epoch 16/20
56/56 [==============================] - 1s 23ms/step - loss: 0.4679 - accura
Epoch 17/20
56/56 [==============================] - 1s 23ms/step - loss: 0.4460 - accura
Epoch 18/20
56/56 [==============================] - 1s 23ms/step - loss: 0.4630 - accura
Epoch 19/20
56/56 [==============================] - 1s 23ms/step - loss: 0.3842 - accura
Epoch 20/20
56/56 [==============================] - 1s 23ms/step - loss: 0.4292 - accura
```

Visualizing training results

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

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Findings

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

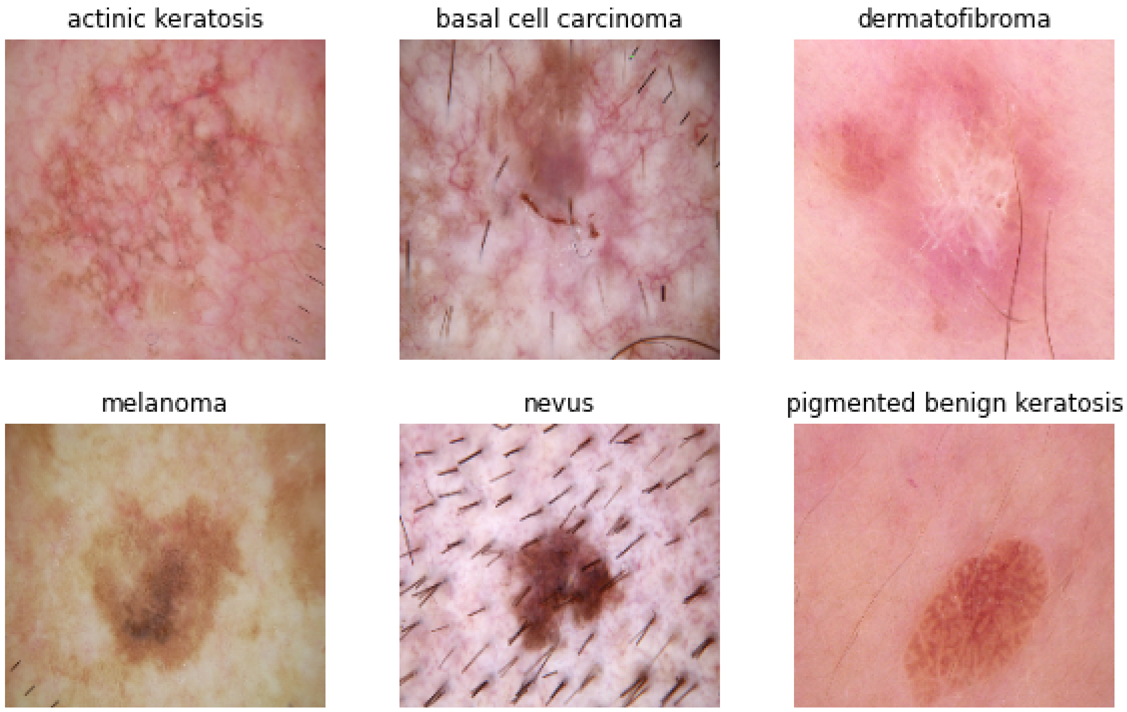
4. Validation Loss : Its decerasing until epoch 10 then increasing per epoch so not a good fit

Model 2: With custom augmentation

```
# Todo, after you have analysed the model fit history for presence of underfit or
# Your code goes here
data_augument = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip(mode="horizontal_and_vertical",in
    layers.experimental.preprocessing.RandomRotation(0.2, fill_mode='reflect'),
    layers.experimental.preprocessing.RandomZoom(height_factor=(0.2, 0.3), width_f
    ])
```

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

# get image each class
class_names_draw = [[] for i in range(len(class_names))]
count = 0
while count < len(class_names):
    for images, labels in train_ds.take(10):
        # augmentation images
        images = data_augument(images)
        for i in range(batch_size):
            class_index = class_names.index(class_names[labels[i]])
            if class_names_draw[class_index] == []:
                class_names_draw[class_index] = images[i].numpy().astype("uint8")
                count +=1


for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(class_names_draw[i])
    plt.title(class_names[i])
    plt.axis("off")
plt.show()
# _ = plt.title(get_label_name(label))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: DeprecationW
```




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

## Your code goes here

model2 = 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)
    ],
    name="Model_02")
model2.summary()
```

```
Model: "Model_02"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_4 (Sequential)   (None, 180, 180, 3)       0

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

 conv2d_15 (Conv2D)          (None, 180, 180, 16)      448

 max_pooling2d_15 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)
```

```
 conv2d_16 (Conv2D)          (None, 90, 90, 32)        4640

 max_pooling2d_16 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 45, 45, 64)        18496

 max_pooling2d_17 (MaxPoolin  (None, 22, 22, 64)       0
 g2D)

 dropout_1 (Dropout)         (None, 22, 22, 64)        0

 flatten_5 (Flatten)         (None, 30976)             0

 dense_10 (Dense)            (None, 128)               3965056

 dense_11 (Dense)            (None, 9)                 1161

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

Compiling the model

```
## Your code goes here
### Todo, choose an appropriate optimiser and loss function
model2.compile(optimizer="adam",
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
              metrics=['accuracy'])
```

Training the model

```
## Your code goes here, note: train your model for 20 epochs
epochs = 30
history = model2.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

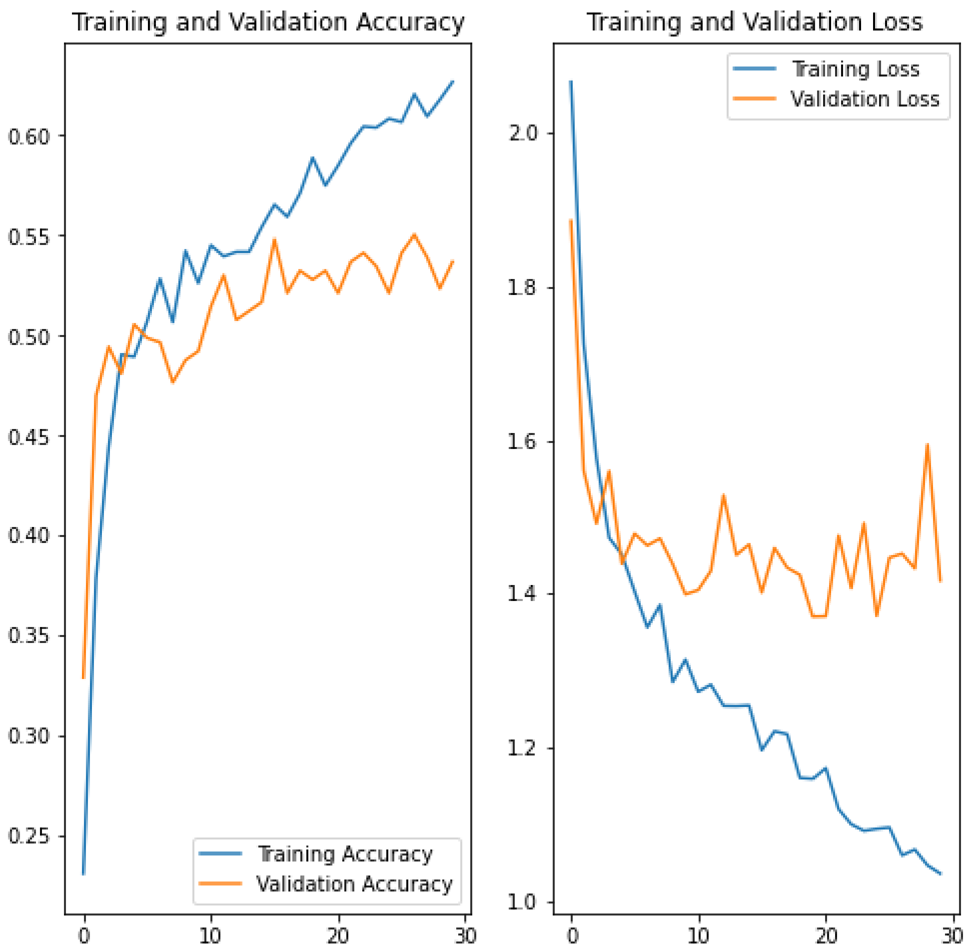
```
56/56 [==============================] - 3s 47ms/step - loss: 1.7251 - accu
Epoch 3/30
56/56 [==============================] - 3s 47ms/step - loss: 1.5766 - accu
Epoch 4/30
56/56 [==============================] - 3s 45ms/step - loss: 1.4726 - accu
Epoch 5/30
56/56 [==============================] - 3s 46ms/step - loss: 1.4507 - accu
Epoch 6/30
56/56 [==============================] - 3s 46ms/step - loss: 1.4026 - accu
Epoch 7/30
56/56 [==============================] - 3s 46ms/step - loss: 1.3557 - accu
Epoch 8/30

56/56 [==============================] - 3s 46ms/step - loss: 1.3850 - accu
Epoch 9/30
56/56 [==============================] - 3s 46ms/step - loss: 1.2844 - accu
```

```
Epoch 10/30
56/56 [==============================] - 3s 46ms/step - loss: 1.3137 - accu
Epoch 11/30
56/56 [==============================] - 3s 47ms/step - loss: 1.2718 - accu
Epoch 12/30
56/56 [==============================] - 3s 45ms/step - loss: 1.2810 - accu
Epoch 13/30
56/56 [==============================] - 3s 46ms/step - loss: 1.2534 - accu
Epoch 14/30
56/56 [==============================] - 3s 45ms/step - loss: 1.2530 - accu
Epoch 15/30
56/56 [==============================] - 3s 45ms/step - loss: 1.2539 - accu
Epoch 16/30
56/56 [==============================] - 3s 46ms/step - loss: 1.1961 - accu
Epoch 17/30
56/56 [==============================] - 3s 47ms/step - loss: 1.2208 - accu
Epoch 18/30
56/56 [==============================] - 3s 47ms/step - loss: 1.2171 - accu
Epoch 19/30
56/56 [==============================] - 3s 45ms/step - loss: 1.1601 - accu
Epoch 20/30
56/56 [==============================] - 3s 45ms/step - loss: 1.1589 - accu
Epoch 21/30
56/56 [==============================] - 3s 46ms/step - loss: 1.1728 - accu
Epoch 22/30
56/56 [==============================] - 3s 45ms/step - loss: 1.1194 - accu
Epoch 23/30
56/56 [==============================] - 3s 46ms/step - loss: 1.0996 - accu
Epoch 24/30
56/56 [==============================] - 3s 45ms/step - loss: 1.0910 - accu
Epoch 25/30
56/56 [==============================] - 3s 46ms/step - loss: 1.0936 - accu
Epoch 26/30
56/56 [==============================] - 3s 46ms/step - loss: 1.0954 - accu
Epoch 27/30
56/56 [==============================] - 4s 71ms/step - loss: 1.0591 - accu
Epoch 28/30
56/56 [==============================] - 3s 45ms/step - loss: 1.0665 - accu
Epoch 29/30
56/56 [==============================] - 3s 46ms/step - loss: 1.0460 - accu
Epoch 30/30
56/56 [==============================] - 3s 45ms/step - loss: 1.0352 - accu
```

Visualizing the results

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

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
```

```
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Findings:

1. The training accuracy go down (underfit), but the distance between train and validation go down to
2. Try to add more layers
3. The analysis before have show that class is imbalanced and ditribution of classes in train and test is different

Model 3: Adding more layers

```
model3 = 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)],
        name="Model_03"
        )
    model3.summary()
```

```
Model: "Model_03"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_4 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 rescaling_9 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_30 (Conv2D)          (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d_30 (MaxPoolin  (None, 90, 90, 16)       0         
 g2D)                                                            
                                                                 
 conv2d_31 (Conv2D)          (None, 90, 90, 32)        4640      
                                                                 
 max_pooling2d_31 (MaxPoolin  (None, 45, 45, 32)       0         
 g2D)                                                            
                                                                 
 dropout_14 (Dropout)        (None, 45, 45, 32)        0         
                                                                 
 conv2d_32 (Conv2D)          (None, 45, 45, 64)        18496     
                                                                 
 max_pooling2d_32 (MaxPoolin  (None, 22, 22, 64)       0         
 g2D)                                                            
                                                                 
 dropout_15 (Dropout)        (None, 22, 22, 64)        0         
                                                                 
 conv2d_33 (Conv2D)          (None, 22, 22, 128)       73856     
                                                                 
 max_pooling2d_33 (MaxPoolin  (None, 11, 11, 128)      0         
 g2D)                                                            
                                                                 
 dropout_16 (Dropout)        (None, 11, 11, 128)       0         
```

```
 flatten_9 (Flatten)         (None, 15488)             0

 dense_18 (Dense)            (None, 128)               1982592

 dropout_17 (Dropout)        (None, 128)               0

 dense_19 (Dense)            (None, 9)                 1161

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

Compile model

```
model3.compile(optimizer="adam",
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
              metrics=['accuracy'])
```

Training the model

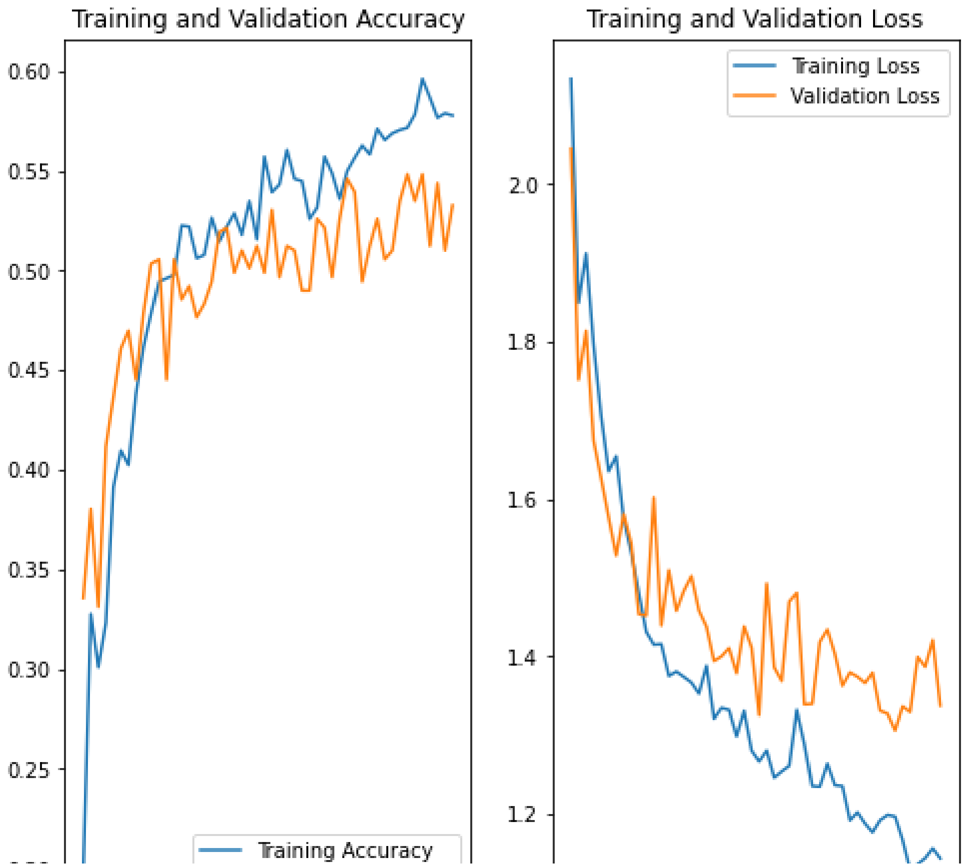
```
epochs = 50
history = model3.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

```
Epoch 22/50
56/56 [==============================] - 3s 48ms/step - loss: 1.3321 - accu
Epoch 23/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2989 - accu
Epoch 24/50
56/56 [==============================] - 3s 49ms/step - loss: 1.3309 - accu
Epoch 25/50
56/56 [==============================] - 3s 49ms/step - loss: 1.2807 - accu
Epoch 26/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2669 - accu
Epoch 27/50
56/56 [==============================] - 3s 49ms/step - loss: 1.2809 - accu
Epoch 28/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2463 - accu
Epoch 29/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2544 - accu
Epoch 30/50
56/56 [==============================] - 3s 47ms/step - loss: 1.2615 - accu
Epoch 31/50
56/56 [==============================] - 3s 48ms/step - loss: 1.3320 - accu
Epoch 32/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2890 - accu
Epoch 33/50
56/56 [==============================] - 3s 47ms/step - loss: 1.2356 - accu
Epoch 34/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2350 - accu
Epoch 35/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2642 - accu
Epoch 36/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2368 - accu
Epoch 37/50
```

```
56/56 [==============================] - 3s 47ms/step - loss: 1.2357 - accu
Epoch 38/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1921 - accu
Epoch 39/50
56/56 [==============================] - 3s 48ms/step - loss: 1.2022 - accu
Epoch 40/50
56/56 [==============================] - 3s 47ms/step - loss: 1.1880 - accu
Epoch 41/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1769 - accu
Epoch 42/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1922 - accu
Epoch 43/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1990 - accu
Epoch 44/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1969 - accu
Epoch 45/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1670 - accu
Epoch 46/50
56/56 [==============================] - 3s 48ms/step - loss: 1.1289 - accu
Epoch 47/50
56/56 [==============================] - 3s 49ms/step - loss: 1.1362 - accu
Epoch 48/50
56/56 [==============================] - 4s 80ms/step - loss: 1.1442 - accu
Epoch 49/50
56/56 [==============================] - 5s 86ms/step - loss: 1.1565 - accu
Epoch 50/50
56/56 [==============================] - 4s 73ms/step - loss: 1.1438 - accu
```

Visualizing the results

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

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Findings

Model has no Overfitting : as both train & validation accuracy move close to overlap (to epoch 30)

Model 4: Adding BatchNorm

```
model4 = 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)],
    name = "Model_04")
```

Compile model

```
model4.compile(optimizer="adam",
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
              metrics=['accuracy'])
```

Training model

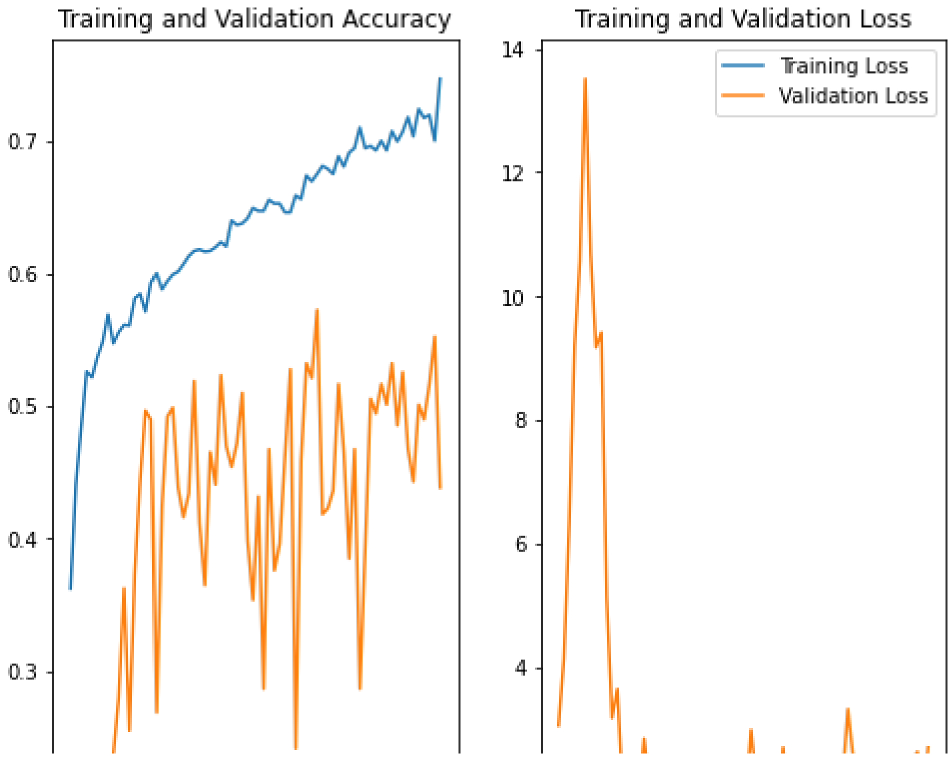
```
## Your code goes here, note: train your model for 20 epochs
epochs = 70
history = model4.fit(train_ds, validation_data=val_ds, epochs=epochs)
```

```
Epoch 42/70
56/56 [==============================] - 3s 53ms/step - loss: 0.9655 - accu
Epoch 43/70
56/56 [==============================] - 3s 52ms/step - loss: 0.9229 - accu
Epoch 44/70
56/56 [==============================] - 3s 51ms/step - loss: 0.9339 - accu
Epoch 45/70
56/56 [==============================] - 3s 50ms/step - loss: 0.8917 - accu
Epoch 46/70
56/56 [==============================] - 3s 51ms/step - loss: 0.8941 - accu
Epoch 47/70
56/56 [==============================] - 3s 52ms/step - loss: 0.8878 - accu
Epoch 48/70
56/56 [==============================] - 3s 52ms/step - loss: 0.8762 - accu
Epoch 49/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8648 - accu
Epoch 50/70
56/56 [==============================] - 3s 52ms/step - loss: 0.8708 - accu
Epoch 51/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8338 - accu
Epoch 52/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8764 - accu
Epoch 53/70
56/56 [==============================] - 3s 52ms/step - loss: 0.8362 - accu
Epoch 54/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8385 - accu
Epoch 55/70
56/56 [==============================] - 3s 51ms/step - loss: 0.8151 - accu
Epoch 56/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8361 - accu
Epoch 57/70
56/56 [==============================] - 3s 56ms/step - loss: 0.8244 - accu
Epoch 58/70
56/56 [==============================] - 4s 71ms/step - loss: 0.8532 - accu
Epoch 59/70
```

```
56/56 [==============================] - 3s 53ms/step - loss: 0.8094 - accu
Epoch 60/70
56/56 [==============================] - 3s 54ms/step - loss: 0.8336 - accu
Epoch 61/70
56/56 [==============================] - 3s 53ms/step - loss: 0.7816 - accu
Epoch 62/70
56/56 [==============================] - 3s 53ms/step - loss: 0.7941 - accu
Epoch 63/70
56/56 [==============================] - 3s 53ms/step - loss: 0.8047 - accu
Epoch 64/70
56/56 [==============================] - 3s 55ms/step - loss: 0.7556 - accu
Epoch 65/70
56/56 [==============================] - 3s 52ms/step - loss: 0.7598 - accu
Epoch 66/70
56/56 [==============================] - 3s 53ms/step - loss: 0.7597 - accu
Epoch 67/70
56/56 [==============================] - 3s 52ms/step - loss: 0.7420 - accu
Epoch 68/70
56/56 [==============================] - 3s 51ms/step - loss: 0.7453 - accu
Epoch 69/70
56/56 [==============================] - 3s 51ms/step - loss: 0.8001 - accu
Epoch 70/70
```

Visualizing the results

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

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Findings

1. No Additional improvement, its due to very less number of data
2. Try improve number of data

Model 5: BatchNorm with more image by Augmentor

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

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Collecting Augmentor
  Downloading Augmentor-0.2.10-py2.py3-none-any.whl (38 kB)
Requirement already satisfied: future>=0.16.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.7/dist
Installing collected packages: Augmentor
Successfully installed Augmentor-0.2.10
```

```
path_to_training_dataset= data_dir_train
import Augmentor
for i in class_names:
    p = Augmentor.Pipeline(os.path.join(path_to_training_dataset, i))
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(1000) ## We are adding 1000 samples per class to make sure that none
```

```
Initialised with 114 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJMU
```

```
Initialised with 376 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 95 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 438 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 357 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 462 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 77 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 181 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
Initialised with 139 image(s) found.
Output directory set to /content/gdrive/MyDrive/LJMU/DeepLearning/Master_LJML
```

Augmentor has stored the augmented images in the output sub-directory of each of the sub-directories of skin cancer types.. Lets take a look at total count of augmented images.

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

```
9000
```

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

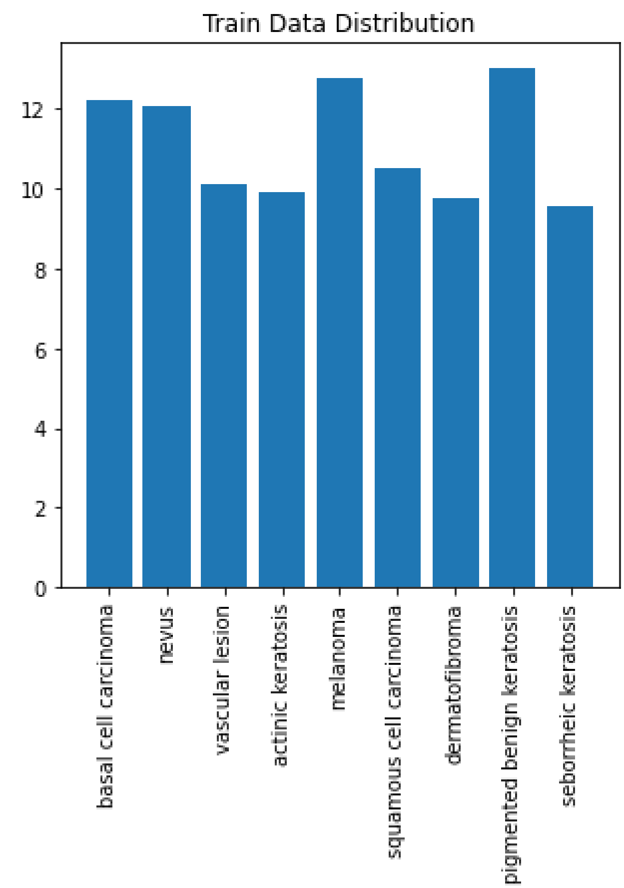
```
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg'))) + len(list(da
print(image_count_train)
```

```
11239
```

```
data_detail_pd = pd.DataFrame(columns=["Dir_Name","Total Image(Train)","Total Perc
# train data in each folders
for dir_name in glob.glob(os.path.join(data_dir_train, "*")):
  total_image_in_folder = len(glob.glob(os.path.join(dir_name, "*.jpg"))) + len(gl
  df = {"Dir_Name":os.path.basename(dir_name),"Total Image(Train)":total_image_in_
  data_detail_pd = data_detail_pd.append(df,ignore_index=True)
data_detail_pd = data_detail_pd.set_index("Dir_Name")
# test data in each folders
for dir_name in glob.glob(os.path.join(data_dir_test, "*")):
  total_image_in_folder = len(glob.glob(os.path.join(dir_name, "*.jpg")))
  data_detail_pd.loc[os.path.basename(dir_name),"Total Image(Test)"]  = total_imag
  data_detail_pd.loc[os.path.basename(dir_name),"Total Percentage(Test)"]  = round
# data_detail_pd = data_detail_pd.set_index("Dir_Name")
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	1462	13.01	16.0	13.56
melanoma	1438	12.79	16.0	13.56
basal cell carcinoma	1376	12.24	16.0	13.56
nevus	1357	12.07	16.0	13.56
squamous cell carcinoma	1181	10.51	16.0	13.56
vascular lesion	1139	10.13	3.0	2.54
actinic keratosis	1114	9.91	16.0	13.56

```
plt.figure(figsize=(5,5))
plt.bar(data_detail_pd.index,data_detail_pd['Total Percentage(Train)'])
plt.xticks(rotation=90)
plt.title("Train Data Distribution")
plt.show()
```



So, now we have added 1000 images to all the classes to maintain some class balance. We can add more images as we want to improve training process

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

Create dataset

```
train_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_train,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(img_height, img_width),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="training",
    interpolation='bicubic',
)
```

```
Found 11239 files belonging to 9 classes.
Using 8992 files for training.
```

```
val_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir_train,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(img_height, img_width),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="validation",
    interpolation='bicubic',
)
```

```
Found 11239 files belonging to 9 classes.
Using 2247 files for validation.
```

Create final model

```
model5 = Sequential([
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
```

```
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(pool_size = (2,2)),
    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(pool_size = (2,2)),
    layers.BatchNormalization(),

    layers.Conv2D(128, 3, padding='same', activation='relu'),

    layers.Conv2D(256, 3, padding='same', activation='relu'),
    #layers.BatchNormalization(),
    #layers.Dropout(0.2),

    layers.Flatten(),

    layers.Dropout(0.2),

    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),

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

    layers.Dense(64, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),

    layers.Dense(32, activation='relu'),
    layers.BatchNormalization(),

    layers.Dense(num_classes)
    ],
    name="Model_05")
model5.summary()
```

```
 conv2d_38 (Conv2D)          (None, 180, 180, 16)      448

 max_pooling2d_38 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

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

 conv2d_39 (Conv2D)          (None, 90, 90, 32)        4640

 conv2d_40 (Conv2D)          (None, 90, 90, 64)        18496

 max_pooling2d_39 (MaxPoolin  (None, 45, 45, 64)       0
```