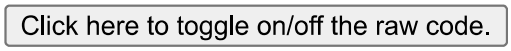
In [374…
```
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here to t
```

Out[374]:



Skin Cancer Data Assignment

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 all the important libraries

In [373…
```
import os
import gc
import PIL
import pathlib
import itertools
from glob import glob

import matplotlib.pyplot as plt
import matplotlib.image as img
import seaborn as sns

import numpy as np
import pandas as pd

from sklearn.metrics import confusion_matrix

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential, Model
from tensorflow.keras.layers import Input, Dense, Layer
from tensorflow.keras.layers import Concatenate, Add
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D, Dro

from tensorflow.keras.utils import plot_model

from albumentations import (Compose, RandomBrightness, RandomBrightnessContrast, J
```

```
                                RandomContrast, HorizontalFlip, Rotate, VerticalFlip)

import logging
logging.getLogger('tensorflow').disabled = True
```

Seed Initilization

In [2]:
```
import random

DEFAULT_RANDOM_SEED = 13

def seedBasic(seed=DEFAULT_RANDOM_SEED):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)

# tensorflow random seed
import tensorflow as tf
def seedTF(seed=DEFAULT_RANDOM_SEED):
    tf.random.set_seed(seed)

# basic + tensorflow
def seedEverything(seed=DEFAULT_RANDOM_SEED):
    seedBasic(seed)
    seedTF(seed)
```

In [3]:
```
seedEverything(DEFAULT_RANDOM_SEED)
```

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.

In [5]:
```
# Defining the path for train and test images
data_dir_train = pathlib.Path("../../data/CNN_assignment/Train")
data_dir_test = pathlib.Path('../../data/CNN_assignment/Test')
```

In [6]:
```
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
```

Loading Data

Let's load these images off disk using the helpful image_dataset_from_directory utility.

Create a Dataset

Define some parameters for the loader:

In [81]:
```
batch_size = 32
img_height = 180
img_width = 180
```

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

In [82]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    validation_split=0.2,
    subset="training",
    seed=DEFAULT_RANDOM_SEED,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

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

In [83]:
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    validation_split=0.2,
    subset="validation",
    seed=DEFAULT_RANDOM_SEED,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

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

In [84]:
```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_test,
    validation_split=0.9999,
    subset="validation",
    seed=DEFAULT_RANDOM_SEED,
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

```
Found 118 files belonging to 9 classes.
Using 117 files for validation.
```

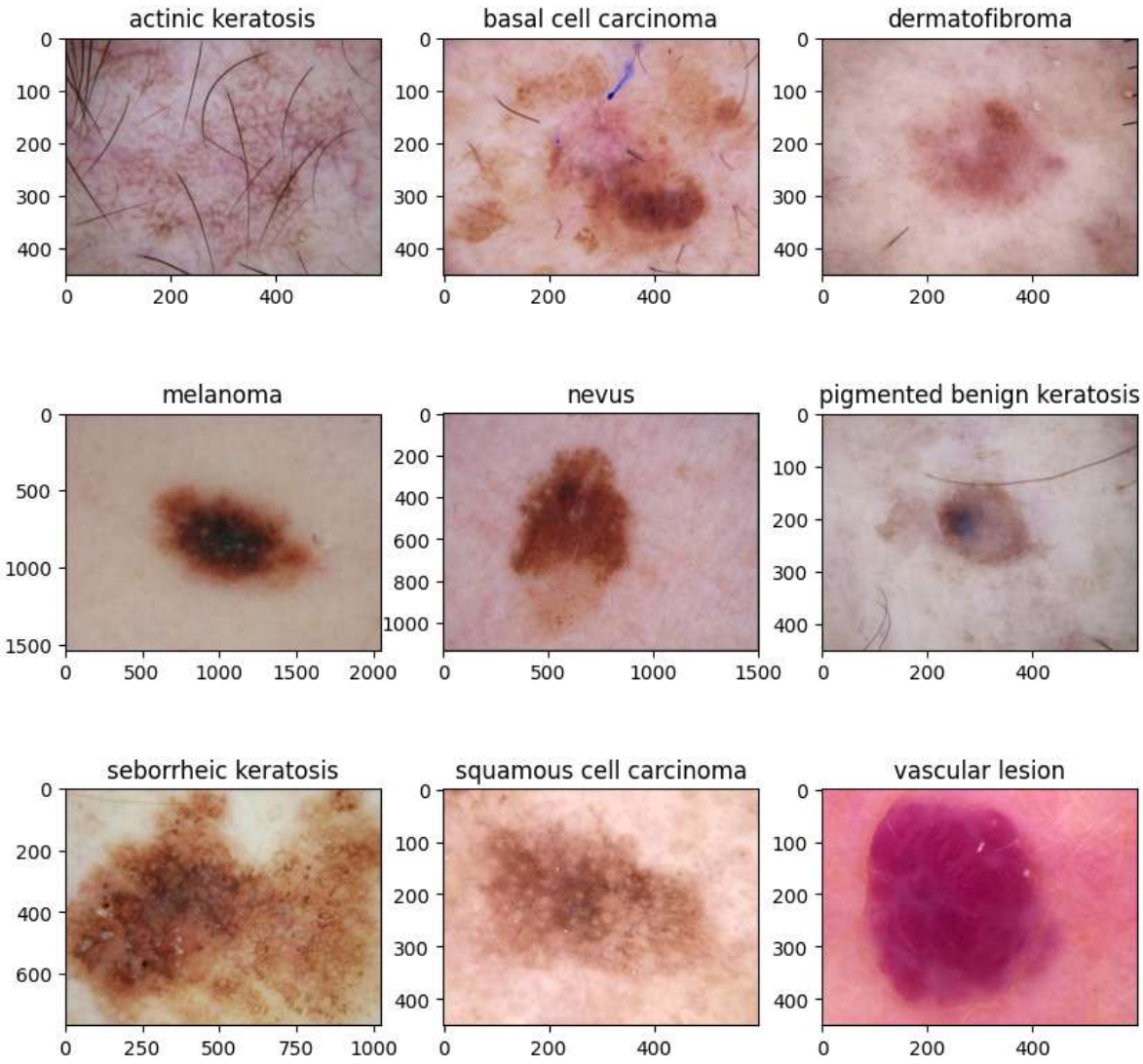
In [85]:
```
class_names = train_ds.class_names
print(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melanoma', 'nevu
s', 'pigmented benign keratosis', 'seborrheic keratosis', 'squamous cell carcinom
a', 'vascular lesion']
```

Visualize the data

In [96]:
```
def visualize_class(class_name):
    return img.imread(str(list(data_dir_train.glob(class_names[class_name]+'/*.jpg
```

In [97]:
```
num_classes = len(class_names)
plt.figure(figsize=(10, 10))
for i in range(num_classes):
    plt.subplot(3, 3, i+1)
    image = visualize_class(i)
    plt.title(class_names[i])
    plt.imshow(image)
```



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.

In [13]:
```
AUTOTUNE = tf.data.experimental.AUTOTUNE

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

Model : SqueezeNet

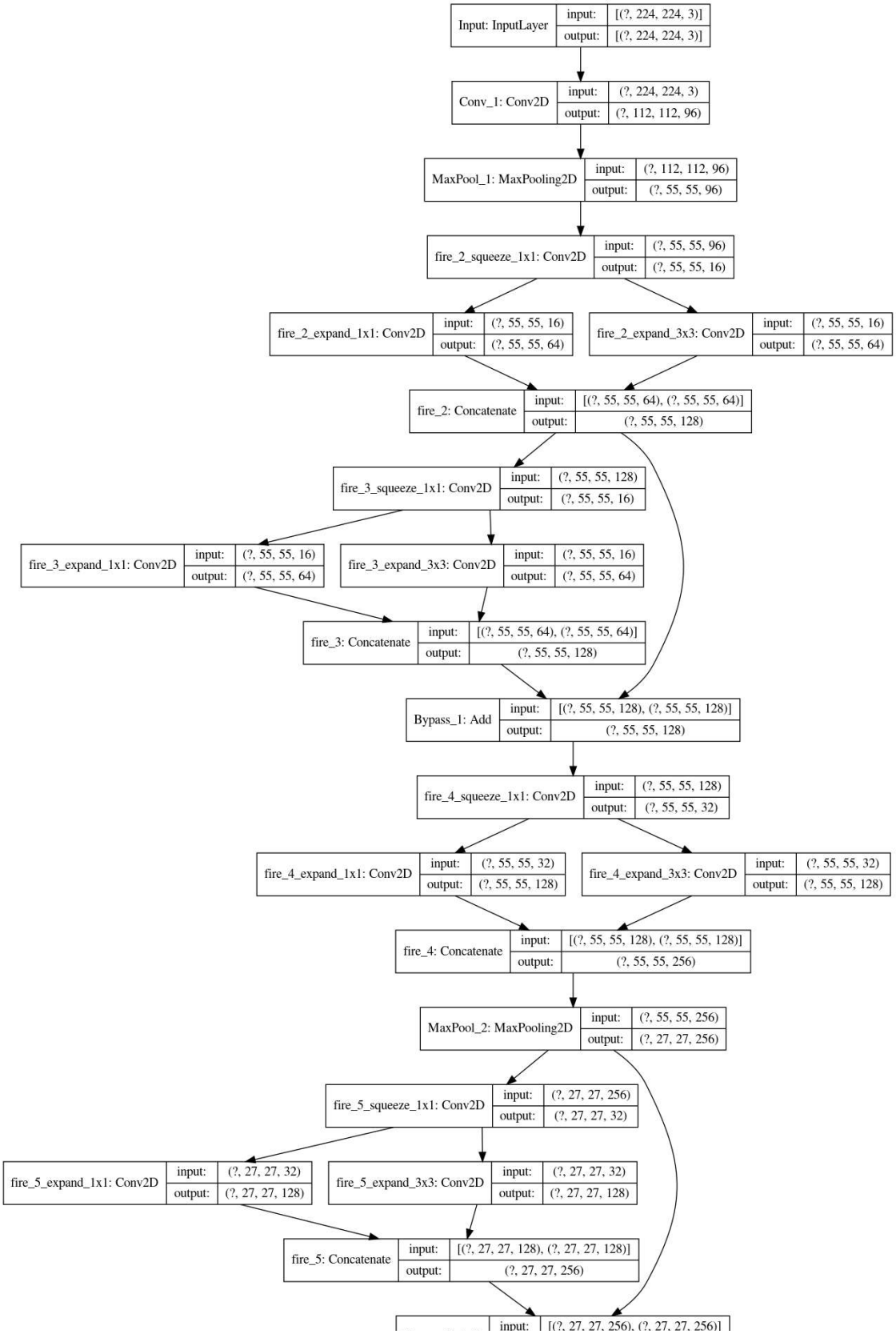
I've built a modified version of SqueezeNet.

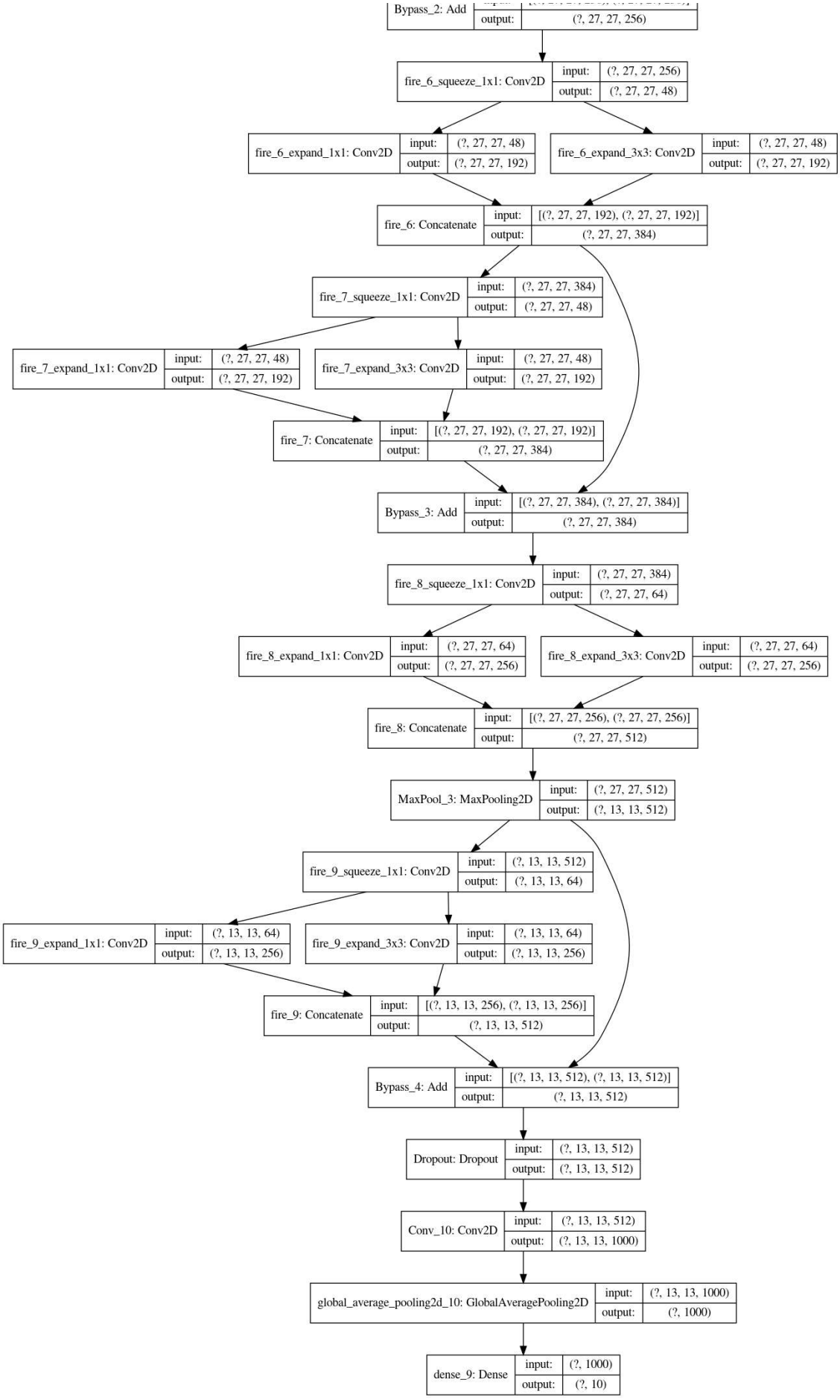
- SqueezeNet : This model is very popular due to its small size and high returns of accuracy per computation just like ENet.
- Why did I choose it :

1. Small Model Size : Can be efficiently deployed in phone cameras and doctors/users can classify the type on the phone itself without visiting the Hospital
2. Faster Training : Lesser Carbon Footprint
3. Very high Return for Accuracy vs Number of Computation

- Modifications : Added BatchNorm after every Conv2d Layers so that training is faster, smoother and better so the number of parameters from the actual SqueezeNet and mine would be a bit different.

Architecture :





Model Skeleton

In [15]:
```
one = (1, 1)
```

```
two = (2, 2)
three = (3, 3)
five = (5, 5)
seven = (7, 7)
thirteen = (13, 13)

input_shape = (224, 224, 3)
```

In [23]:

```
class FireModule(object):
    """
    Fire Module computed as per the SqueezeNet paper
    """

    def __init__(self, layer_number: int, activation: str, kernel_initializer: str
        """
        Constructor

        Arguments:
          layer_number       : Index of the Fire Module
          activation         : Activation to be used
          kernel_initializer : Kernel Weight Initialization technique

        Returns:
          None
        """

        self.layer_number = layer_number
        self.activation = activation
        self.kernel_initializer = kernel_initializer

    def build_module(self, fire_input: Layer) -> Layer:
        """
        Build the SqueezeNet

        Arguments:
          fire_input       : Input to Fire Module

        Returns:
          model            : SqueezeNet
        """

        global one, three, five

        output_size = 128 * (1 + (self.layer_number//2))

        squeeze_1x1_filters = 16 * (1 + (self.layer_number//2))
        expand_1x1_filters = expand_3x3_filters = output_size//2

        squeeze_1x1 = Conv2D(name=f'fire_{self.layer_number+2}_squeeze_1x1',
            filters=squeeze_1x1_filters, kernel_size=one, strides=1, padding='vali
            kernel_initializer=self.kernel_initializer)(fire_input)
        bn_1 = BatchNormalization(momentum=0.9)(squeeze_1x1)
        expand_1x1 = Conv2D(name=f'fire_{self.layer_number+2}_expand_1x1',
            filters=expand_1x1_filters, kernel_size=one, strides=1, padding='valid
            kernel_initializer=self.kernel_initializer)(bn_1)
        bn_2 = BatchNormalization(momentum=0.9)(expand_1x1)
        expand_3x3 = Conv2D(name=f'fire_{self.layer_number+2}_expand_3x3',
            filters=expand_3x3_filters, kernel_size=three, strides=1, padding='same
            kernel_initializer=self.kernel_initializer)(bn_2)
        bn_3 = BatchNormalization(momentum=0.9)(expand_3x3)

        fire = Concatenate(name=f'fire_{self.layer_number+2}')([bn_2, bn_3])
```

```
        return fire


class SqueezeNet(object):
    """
    SqueezeNet Architecture
    """

    def __init__(self, activation: str='relu', kernel_initializer: str='glorot_uni
        """
        Constructor

        Arguments:
          activation         : Activation to be used
          kernel_initializer : Kernel Weight Initialization technique

        Returns:
          None
        """

        self.activation = activation
        self.kernel_initializer = kernel_initializer

    def vanilla_model(self, input_shape: tuple=(224, 224, 3), n_classes: int=1000)
        """
        Vanilla Implementation of SqueezeNet

        Arguments:
          input_shape         : Input Shape of the images
          n_classes           : Number of output classes

        Returns:
          None
        """

        inp = Input(shape=input_shape, name='Input')

        # Conv1 Layer
        conv_1 = Conv2D(name="Conv_1",
            filters=96, kernel_size=seven, strides=2, padding='same', activation=se
        bn_1 = BatchNormalization(momentum=0.9)(conv_1)
        maxpool_1 = MaxPool2D(name="MaxPool_1",
            pool_size=three, strides=2)(bn_1)

        # Fire 2-4
        fire_2 = FireModule(layer_number=0, activation=self.activation,
                            kernel_initializer=self.kernel_initializer).build_modul
        fire_3 = FireModule(layer_number=1, activation=self.activation,
                            kernel_initializer=self.kernel_initializer).build_modul
        fire_4 = FireModule(layer_number=2, activation=self.activation,
                            kernel_initializer=self.kernel_initializer).build_modul

        # Max Pool after Fire4 Module
        maxpool_2 = MaxPool2D(name="MaxPool_2",
            pool_size=three, strides=2)(fire_4)

        # Fire 5-8
        fire_5 = FireModule(layer_number=3, activation=self.activation, kernel_init
        fire_6 = FireModule(layer_number=4, activation=self.activation, kernel_init
        fire_7 = FireModule(layer_number=5, activation=self.activation, kernel_init
        fire_8 = FireModule(layer_number=6, activation=self.activation, kernel_init

        # Max Pool after Fire8 Module
        maxpool_3 = MaxPool2D(name="MaxPool_3",
```

```
            pool_size=three, strides=2)(fire_8)

        fire_9 = FireModule(layer_number=7, activation=self.activation, kernel_init

        # Dropout
        dropout = Dropout(0.5, name="Dropout")(fire_9)

        # Conv10 Layer
        conv_10 = Conv2D(name="Conv_10",
            filters=1000, kernel_size=one, strides=1, padding='valid', activation=
        bn_2 = BatchNormalization(momentum=0.9)(conv_10)
        gap_11 = GlobalAveragePooling2D()(bn_2)


        if n_classes != 1000:
            # Add Dense(n_classes) and ouput == Dense layer
            out = Dense(n_classes, activation='softmax')(gap_11)
        else:
            out = gap_11

        self.model = Model(inputs=inp, outputs=out)


    def bypass_model(self, input_shape: tuple=(224, 224, 3), n_classes: int=1000)
        """
        Residual Inspired Bypass Implementation of SqueezeNet

        Arguments:
          input_shape         : Input Shape of the images
          n_classes           : Number of output classes

        Returns:
          None
        """

        inp = Input(shape=input_shape, name='Input')

        # Conv1 Layer
        conv_1 = Conv2D(name="Conv_1",
            filters=96, kernel_size=seven, strides=2, padding='same', activation=s
        bn_1 = BatchNormalization(momentum=0.9)(conv_1)
        maxpool_1 = MaxPool2D(name="MaxPool_1",
            pool_size=three, strides=2)(bn_1)

        # Fire 2-4
        fire_2 = FireModule(layer_number=0, activation=self.activation, kernel_init
        fire_3 = FireModule(layer_number=1, activation=self.activation, kernel_init
        bypass_1 = Add(name="Bypass_1")([fire_2, fire_3])
        fire_4 = FireModule(layer_number=2, activation=self.activation, kernel_init

        # Max Pool after Fire4 Module
        maxpool_2 = MaxPool2D(name="MaxPool_2",
            pool_size=three, strides=2)(fire_4)

        # Fire 5-8
        fire_5 = FireModule(layer_number=3, activation=self.activation, kernel_init
        bypass_2 = Add(name="Bypass_2")([maxpool_2, fire_5])
        fire_6 = FireModule(layer_number=4, activation=self.activation, kernel_init
        fire_7 = FireModule(layer_number=5, activation=self.activation, kernel_init
        bypass_3 = Add(name="Bypass_3")([fire_6, fire_7])
        fire_8 = FireModule(layer_number=6, activation=self.activation, kernel_init

        # Max Pool after Fire8 Module
        maxpool_3 = MaxPool2D(name="MaxPool_3",
```

```
            pool_size=three, strides=2)(fire_8)

        fire_9 = FireModule(layer_number=7, activation=self.activation, kernel_ini
        bypass_4 = Add(name="Bypass_4")([maxpool_3, fire_9])

        # Dropout
        dropout = Dropout(0.5, name="Dropout")(bypass_4)

        # Conv10 layer
        conv_10 = Conv2D(name="Conv_10",
            filters=1000, kernel_size=one, strides=1, padding='valid', activation=
        bn_2 = BatchNormalization(momentum=0.9)(conv_10)
        gap_11 = GlobalAveragePooling2D()(bn_2)

#         fcn = Dense(256, activation='relu')(gap_11)

        if n_classes != 1000:
            out = Dense(n_classes, activation='softmax')(gap_11)
        else:
            out = gap_11

        self.model = Model(inputs=inp, outputs=out)


    def build_model(self, input_shape: tuple=(224, 224, 3), n_classes: int=1000, c
        """
        Build SqueezeNet

        Arguments:
          input_shape         : Input Shape of the images
          n_classes           : Number of output classes
          choice              : Type of architecture (vanilla/bypass)
        Returns:
          model               : SqueezeNet Model
        """

        if choice == "vanilla":
            self.vanilla_model(input_shape, n_classes)
        else:
            self.bypass_model(input_shape, n_classes)

        return self.model
```

In [86]:

```
snet = SqueezeNet()

model = snet.build_model(n_classes=9, choice='bypass', input_shape=(180, 180, 3))
model.summary()
```

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 Input (InputLayer)          [(None, 180, 180, 3       0         []
                             )]

 Conv_1 (Conv2D)             (None, 90, 90, 96)        14208     ['Input[0][0]']

 batch_normalization_130 (Batch  (None, 90, 90, 96)    384       ['Conv_1[0][0]']
 Normalization)

 MaxPool_1 (MaxPooling2D)    (None, 44, 44, 96)        0         ['batch_normaliza
tion_130[0][0]']

 fire_2_squeeze_1x1 (Conv2D)  (None, 44, 44, 16)       1552      ['MaxPool_1[0]
[0]']

 batch_normalization_131 (Batch  (None, 44, 44, 16)    64        ['fire_2_squeeze_
1x1[0][0]']
 Normalization)

 fire_2_expand_1x1 (Conv2D)  (None, 44, 44, 64)        1088      ['batch_normaliza
tion_131[0][0]']

 batch_normalization_132 (Batch  (None, 44, 44, 64)    256       ['fire_2_expand_1
x1[0][0]']
 Normalization)

 fire_2_expand_3x3 (Conv2D)  (None, 44, 44, 64)        36928     ['batch_normaliza
tion_132[0][0]']

 batch_normalization_133 (Batch  (None, 44, 44, 64)    256       ['fire_2_expand_3
x3[0][0]']
 Normalization)

 fire_2 (Concatenate)        (None, 44, 44, 128)       0         ['batch_normaliza
tion_132[0][0]',
                                                                  'batch_normaliza
tion_133[0][0]']

 fire_3_squeeze_1x1 (Conv2D)  (None, 44, 44, 16)       2064      ['fire_2[0][0]']

 batch_normalization_134 (Batch  (None, 44, 44, 16)    64        ['fire_3_squeeze_
1x1[0][0]']
 Normalization)

 fire_3_expand_1x1 (Conv2D)  (None, 44, 44, 64)        1088      ['batch_normaliza
tion_134[0][0]']

 batch_normalization_135 (Batch  (None, 44, 44, 64)    256       ['fire_3_expand_1
x1[0][0]']
 Normalization)

 fire_3_expand_3x3 (Conv2D)  (None, 44, 44, 64)        36928     ['batch_normaliza
tion_135[0][0]']

 batch_normalization_136 (Batch  (None, 44, 44, 64)    256       ['fire_3_expand_3
x3[0][0]']
 Normalization)

 fire_3 (Concatenate)        (None, 44, 44, 128)       0         ['batch_normaliza
```

```
tion_135[0][0]',
                                                                  'batch_normaliza
tion_136[0][0]']

 Bypass_1 (Add)                 (None, 44, 44, 128)  0           ['fire_2[0][0]',
                                                                  'fire_3[0][0]']

 fire_4_squeeze_1x1 (Conv2D)    (None, 44, 44, 32)   4128        ['Bypass_1[0]
[0]']

 batch_normalization_137 (Batch  (None, 44, 44, 32)  128         ['fire_4_squeeze_
1x1[0][0]']
 Normalization)

 fire_4_expand_1x1 (Conv2D)     (None, 44, 44, 128)  4224        ['batch_normaliza
tion_137[0][0]']

 batch_normalization_138 (Batch  (None, 44, 44, 128)  512        ['fire_4_expand_1
x1[0][0]']
 Normalization)

 fire_4_expand_3x3 (Conv2D)     (None, 44, 44, 128)  147584      ['batch_normaliza
tion_138[0][0]']

 batch_normalization_139 (Batch  (None, 44, 44, 128)  512        ['fire_4_expand_3
x3[0][0]']
 Normalization)

 fire_4 (Concatenate)           (None, 44, 44, 256)  0           ['batch_normaliza
tion_138[0][0]',
                                                                  'batch_normaliza
tion_139[0][0]']

 MaxPool_2 (MaxPooling2D)       (None, 21, 21, 256)  0           ['fire_4[0][0]']

 fire_5_squeeze_1x1 (Conv2D)    (None, 21, 21, 32)   8224        ['MaxPool_2[0]
[0]']

 batch_normalization_140 (Batch  (None, 21, 21, 32)  128         ['fire_5_squeeze_
1x1[0][0]']
 Normalization)

 fire_5_expand_1x1 (Conv2D)     (None, 21, 21, 128)  4224        ['batch_normaliza
tion_140[0][0]']

 batch_normalization_141 (Batch  (None, 21, 21, 128)  512        ['fire_5_expand_1
x1[0][0]']
 Normalization)

 fire_5_expand_3x3 (Conv2D)     (None, 21, 21, 128)  147584      ['batch_normaliza
tion_141[0][0]']

 batch_normalization_142 (Batch  (None, 21, 21, 128)  512        ['fire_5_expand_3
x3[0][0]']
 Normalization)

 fire_5 (Concatenate)           (None, 21, 21, 256)  0           ['batch_normaliza
tion_141[0][0]',
                                                                  'batch_normaliza
tion_142[0][0]']

 Bypass_2 (Add)                 (None, 21, 21, 256)  0           ['MaxPool_2[0]
[0]',
                                                                  'fire_5[0][0]']
```

```
 fire_6_squeeze_1x1 (Conv2D)    (None, 21, 21, 48)   12336       ['Bypass_2[0]
[0]']

 batch_normalization_143 (Batch  (None, 21, 21, 48)  192         ['fire_6_squeeze_
1x1[0][0]']
 Normalization)

 fire_6_expand_1x1 (Conv2D)     (None, 21, 21, 192)  9408        ['batch_normaliza
tion_143[0][0]']

 batch_normalization_144 (Batch  (None, 21, 21, 192)  768        ['fire_6_expand_1
x1[0][0]']
 Normalization)

 fire_6_expand_3x3 (Conv2D)     (None, 21, 21, 192)  331968      ['batch_normaliza
tion_144[0][0]']

 batch_normalization_145 (Batch  (None, 21, 21, 192)  768        ['fire_6_expand_3
x3[0][0]']
 Normalization)

 fire_6 (Concatenate)           (None, 21, 21, 384)  0           ['batch_normaliza
tion_144[0][0]',
                                                                  'batch_normaliza
tion_145[0][0]']

 fire_7_squeeze_1x1 (Conv2D)    (None, 21, 21, 48)   18480       ['fire_6[0][0]']

 batch_normalization_146 (Batch  (None, 21, 21, 48)  192         ['fire_7_squeeze_
1x1[0][0]']
 Normalization)

 fire_7_expand_1x1 (Conv2D)     (None, 21, 21, 192)  9408        ['batch_normaliza
tion_146[0][0]']

 batch_normalization_147 (Batch  (None, 21, 21, 192)  768        ['fire_7_expand_1
x1[0][0]']
 Normalization)

 fire_7_expand_3x3 (Conv2D)     (None, 21, 21, 192)  331968      ['batch_normaliza
tion_147[0][0]']

 batch_normalization_148 (Batch  (None, 21, 21, 192)  768        ['fire_7_expand_3
x3[0][0]']
 Normalization)

 fire_7 (Concatenate)           (None, 21, 21, 384)  0           ['batch_normaliza
tion_147[0][0]',
                                                                  'batch_normaliza
tion_148[0][0]']

 Bypass_3 (Add)                 (None, 21, 21, 384)  0           ['fire_6[0][0]',
                                                                  'fire_7[0][0]']

 fire_8_squeeze_1x1 (Conv2D)    (None, 21, 21, 64)   24640       ['Bypass_3[0]
[0]']

 batch_normalization_149 (Batch  (None, 21, 21, 64)  256         ['fire_8_squeeze_
1x1[0][0]']
 Normalization)

 fire_8_expand_1x1 (Conv2D)     (None, 21, 21, 256)  16640       ['batch_normaliza
tion_149[0][0]']
```

```
 batch_normalization_150 (Batch  (None, 21, 21, 256)  1024       ['fire_8_expand_1
x1[0][0]']
 Normalization)

 fire_8_expand_3x3 (Conv2D)     (None, 21, 21, 256)  590080      ['batch_normaliza
tion_150[0][0]']

 batch_normalization_151 (Batch  (None, 21, 21, 256)  1024       ['fire_8_expand_3
x3[0][0]']
 Normalization)

 fire_8 (Concatenate)           (None, 21, 21, 512)  0           ['batch_normaliza
tion_150[0][0]',
                                                                  'batch_normaliza
tion_151[0][0]']

 MaxPool_3 (MaxPooling2D)       (None, 10, 10, 512)  0           ['fire_8[0][0]']

 fire_9_squeeze_1x1 (Conv2D)    (None, 10, 10, 64)   32832       ['MaxPool_3[0]
[0]']

 batch_normalization_152 (Batch  (None, 10, 10, 64)  256         ['fire_9_squeeze_
1x1[0][0]']
 Normalization)

 fire_9_expand_1x1 (Conv2D)     (None, 10, 10, 256)  16640       ['batch_normaliza
tion_152[0][0]']

 batch_normalization_153 (Batch  (None, 10, 10, 256)  1024       ['fire_9_expand_1
x1[0][0]']
 Normalization)

 fire_9_expand_3x3 (Conv2D)     (None, 10, 10, 256)  590080      ['batch_normaliza
tion_153[0][0]']

 batch_normalization_154 (Batch  (None, 10, 10, 256)  1024       ['fire_9_expand_3
x3[0][0]']
 Normalization)

 fire_9 (Concatenate)           (None, 10, 10, 512)  0           ['batch_normaliza
tion_153[0][0]',
                                                                  'batch_normaliza
tion_154[0][0]']

 Bypass_4 (Add)                 (None, 10, 10, 512)  0           ['MaxPool_3[0]
[0]',
                                                                  'fire_9[0][0]']

 Dropout (Dropout)              (None, 10, 10, 512)  0           ['Bypass_4[0]
[0]']

 Conv_10 (Conv2D)               (None, 10, 10, 1000  513000      ['Dropout[0][0]']
                                )

 batch_normalization_155 (Batch  (None, 10, 10, 1000  4000       ['Conv_10[0][0]']
 Normalization)                 )

 global_average_pooling2d_6 (Gl  (None, 1000)        0           ['batch_normaliza
tion_155[0][0]']
 obalAveragePooling2D)

 dense_6 (Dense)                (None, 9)            9009        ['global_average_
pooling2d_6[0][0
```

```
']
```

```
================================================================================
================
Total params: 2,932,217
Trainable params: 2,924,265
Non-trainable params: 7,952
_________________________________________________________________
```

Compile the model

- Optimizer : Choosing Adam as the Optimizer as its very widely used and is efficient and fast!
- Loss : SparseCategoricalCrossEntropy because we have a multiclass classification problem at hand.
- Metric : Accuracy

In [88]:
```
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer= opt,
              loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Fals
              metrics=['accuracy'])
```

Training the model

In [49]:
```
es_cb = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=10, rest
rdr_cb = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, pati
ckpt_cb = tf.keras.callbacks.ModelCheckpoint(filepath="baseline_snet_weights.hdf5"
```

In [89]:
```
epochs = 40
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    callbacks=[es_cb, rdr_cb, ckpt_cb],
)
```

```
Epoch 1/100
56/56 [==============================] - ETA: 0s - loss: 1.9659 - accuracy: 0.3387
Epoch 1: val_accuracy did not improve from 0.55928
56/56 [==============================] - 16s 212ms/step - loss: 1.9659 - accuracy:
0.3387 - val_loss: 2.0818 - val_accuracy: 0.3199 - lr: 0.0010
Epoch 2/100
56/56 [==============================] - ETA: 0s - loss: 1.6830 - accuracy: 0.4107
Epoch 2: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.6830 - accuracy:
0.4107 - val_loss: 1.9436 - val_accuracy: 0.3602 - lr: 0.0010
Epoch 3/100
56/56 [==============================] - ETA: 0s - loss: 1.5738 - accuracy: 0.4392
Epoch 3: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.5738 - accuracy:
0.4392 - val_loss: 1.5542 - val_accuracy: 0.4541 - lr: 0.0010
Epoch 4/100
56/56 [==============================] - ETA: 0s - loss: 1.4451 - accuracy: 0.4760
Epoch 4: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 198ms/step - loss: 1.4451 - accuracy:
0.4760 - val_loss: 1.8718 - val_accuracy: 0.3311 - lr: 0.0010
Epoch 5/100
56/56 [==============================] - ETA: 0s - loss: 1.4097 - accuracy: 0.5022
Epoch 5: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 198ms/step - loss: 1.4097 - accuracy:
0.5022 - val_loss: 1.6511 - val_accuracy: 0.4452 - lr: 0.0010
Epoch 6/100
56/56 [==============================] - ETA: 0s - loss: 1.3515 - accuracy: 0.5184
Epoch 6: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.3515 - accuracy:
0.5184 - val_loss: 1.8792 - val_accuracy: 0.3848 - lr: 0.0010
Epoch 7/100
56/56 [==============================] - ETA: 0s - loss: 1.3367 - accuracy: 0.5151
Epoch 7: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.3367 - accuracy:
0.5151 - val_loss: 1.4518 - val_accuracy: 0.4899 - lr: 0.0010
Epoch 8/100
56/56 [==============================] - ETA: 0s - loss: 1.2838 - accuracy: 0.5385
Epoch 8: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 200ms/step - loss: 1.2838 - accuracy:
0.5385 - val_loss: 1.5630 - val_accuracy: 0.5034 - lr: 0.0010
Epoch 9/100
56/56 [==============================] - ETA: 0s - loss: 1.2641 - accuracy: 0.5547
Epoch 9: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 198ms/step - loss: 1.2641 - accuracy:
0.5547 - val_loss: 1.3910 - val_accuracy: 0.5034 - lr: 0.0010
Epoch 10/100
56/56 [==============================] - ETA: 0s - loss: 1.2664 - accuracy: 0.5525
Epoch 10: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.2664 - accuracy:
0.5525 - val_loss: 1.6346 - val_accuracy: 0.4206 - lr: 0.0010
Epoch 11/100
56/56 [==============================] - ETA: 0s - loss: 1.2187 - accuracy: 0.5547
Epoch 11: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.2187 - accuracy:
0.5547 - val_loss: 1.5013 - val_accuracy: 0.4362 - lr: 0.0010
Epoch 12/100
56/56 [==============================] - ETA: 0s - loss: 1.2011 - accuracy: 0.5692
Epoch 12: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.2011 - accuracy:
0.5692 - val_loss: 1.5914 - val_accuracy: 0.4720 - lr: 0.0010
Epoch 13/100
56/56 [==============================] - ETA: 0s - loss: 1.2114 - accuracy: 0.5681
Epoch 13: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 200ms/step - loss: 1.2114 - accuracy:
```

```
0.5681 - val_loss: 1.5033 - val_accuracy: 0.4474 - lr: 0.0010
Epoch 14/100
56/56 [==============================] - ETA: 0s - loss: 1.1590 - accuracy: 0.5871
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 14: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 199ms/step - loss: 1.1590 - accuracy:
0.5871 - val_loss: 1.5602 - val_accuracy: 0.4765 - lr: 0.0010
Epoch 15/100
56/56 [==============================] - ETA: 0s - loss: 1.0700 - accuracy: 0.6071
Epoch 15: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 200ms/step - loss: 1.0700 - accuracy:
0.6071 - val_loss: 1.2802 - val_accuracy: 0.5391 - lr: 5.0000e-04
Epoch 16/100
56/56 [==============================] - ETA: 0s - loss: 1.0383 - accuracy: 0.6278
Epoch 16: val_accuracy did not improve from 0.55928
56/56 [==============================] - 12s 201ms/step - loss: 1.0383 - accuracy:
0.6278 - val_loss: 1.3074 - val_accuracy: 0.5459 - lr: 5.0000e-04
Epoch 17/100
56/56 [==============================] - ETA: 0s - loss: 0.9941 - accuracy: 0.6490
Epoch 17: val_accuracy improved from 0.55928 to 0.57271, saving model to weights-
0.57.hdf5
56/56 [==============================] - 12s 207ms/step - loss: 0.9941 - accuracy:
0.6490 - val_loss: 1.1998 - val_accuracy: 0.5727 - lr: 5.0000e-04
Epoch 18/100
56/56 [==============================] - ETA: 0s - loss: 0.9822 - accuracy: 0.6574
Epoch 18: val_accuracy did not improve from 0.57271
56/56 [==============================] - 12s 199ms/step - loss: 0.9822 - accuracy:
0.6574 - val_loss: 1.5188 - val_accuracy: 0.4966 - lr: 5.0000e-04
Epoch 19/100
56/56 [==============================] - ETA: 0s - loss: 0.9539 - accuracy: 0.6641
Epoch 19: val_accuracy did not improve from 0.57271
56/56 [==============================] - 12s 201ms/step - loss: 0.9539 - accuracy:
0.6641 - val_loss: 1.2737 - val_accuracy: 0.5593 - lr: 5.0000e-04
Epoch 20/100
56/56 [==============================] - ETA: 0s - loss: 0.9302 - accuracy: 0.6669
Epoch 20: val_accuracy did not improve from 0.57271
56/56 [==============================] - 12s 199ms/step - loss: 0.9302 - accuracy:
0.6669 - val_loss: 1.3975 - val_accuracy: 0.5235 - lr: 5.0000e-04
Epoch 21/100
56/56 [==============================] - ETA: 0s - loss: 0.8980 - accuracy: 0.6819
Epoch 21: val_accuracy improved from 0.57271 to 0.58166, saving model to weights-
0.58.hdf5
56/56 [==============================] - 12s 209ms/step - loss: 0.8980 - accuracy:
0.6819 - val_loss: 1.3480 - val_accuracy: 0.5817 - lr: 5.0000e-04
Epoch 22/100
56/56 [==============================] - ETA: 0s - loss: 0.9056 - accuracy: 0.6691
Epoch 22: val_accuracy did not improve from 0.58166
56/56 [==============================] - 12s 199ms/step - loss: 0.9056 - accuracy:
0.6691 - val_loss: 1.6460 - val_accuracy: 0.4743 - lr: 5.0000e-04
Epoch 23/100
56/56 [==============================] - ETA: 0s - loss: 0.8680 - accuracy: 0.6925
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.

Epoch 23: val_accuracy did not improve from 0.58166
56/56 [==============================] - 12s 198ms/step - loss: 0.8680 - accuracy:
0.6925 - val_loss: 1.5125 - val_accuracy: 0.5190 - lr: 5.0000e-04
Epoch 24/100
56/56 [==============================] - ETA: 0s - loss: 0.7774 - accuracy: 0.7143
Epoch 24: val_accuracy did not improve from 0.58166
56/56 [==============================] - 12s 200ms/step - loss: 0.7774 - accuracy:
0.7143 - val_loss: 1.3309 - val_accuracy: 0.5749 - lr: 2.5000e-04
Epoch 25/100
56/56 [==============================] - ETA: 0s - loss: 0.7245 - accuracy: 0.7455
```

```
Epoch 25: val_accuracy improved from 0.58166 to 0.58837, saving model to weights-
0.59.hdf5
56/56 [==============================] - 12s 208ms/step - loss: 0.7245 - accuracy:
0.7455 - val_loss: 1.2035 - val_accuracy: 0.5884 - lr: 2.5000e-04
Epoch 26/100
56/56 [==============================] - ETA: 0s - loss: 0.6704 - accuracy: 0.7528
Epoch 26: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.6704 - accuracy:
0.7528 - val_loss: 1.3273 - val_accuracy: 0.5682 - lr: 2.5000e-04
Epoch 27/100
56/56 [==============================] - ETA: 0s - loss: 0.6658 - accuracy: 0.7561
Epoch 27: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.6658 - accuracy:
0.7561 - val_loss: 1.2125 - val_accuracy: 0.5884 - lr: 2.5000e-04
Epoch 28/100
56/56 [==============================] - ETA: 0s - loss: 0.6116 - accuracy: 0.7857
Epoch 28: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.6116 - accuracy:
0.7857 - val_loss: 1.3762 - val_accuracy: 0.5772 - lr: 2.5000e-04
Epoch 29/100
56/56 [==============================] - ETA: 0s - loss: 0.5827 - accuracy: 0.7935
Epoch 29: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.5827 - accuracy:
0.7935 - val_loss: 1.5328 - val_accuracy: 0.5638 - lr: 2.5000e-04
Epoch 30/100
56/56 [==============================] - ETA: 0s - loss: 0.5863 - accuracy: 0.7930
Epoch 30: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 197ms/step - loss: 0.5863 - accuracy:
0.7930 - val_loss: 1.5082 - val_accuracy: 0.5817 - lr: 2.5000e-04
Epoch 31/100
56/56 [==============================] - ETA: 0s - loss: 0.5596 - accuracy: 0.7863
Epoch 31: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 196ms/step - loss: 0.5596 - accuracy:
0.7863 - val_loss: 1.4870 - val_accuracy: 0.5772 - lr: 2.5000e-04
Epoch 32/100
56/56 [==============================] - ETA: 0s - loss: 0.5682 - accuracy: 0.7985
Epoch 32: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.

Epoch 32: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 197ms/step - loss: 0.5682 - accuracy:
0.7985 - val_loss: 1.6783 - val_accuracy: 0.5257 - lr: 2.5000e-04
Epoch 33/100
56/56 [==============================] - ETA: 0s - loss: 0.4536 - accuracy: 0.8242
Epoch 33: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.4536 - accuracy:
0.8242 - val_loss: 1.3997 - val_accuracy: 0.5705 - lr: 1.2500e-04
Epoch 34/100
56/56 [==============================] - ETA: 0s - loss: 0.3913 - accuracy: 0.8605
Epoch 34: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 199ms/step - loss: 0.3913 - accuracy:
0.8605 - val_loss: 1.4496 - val_accuracy: 0.5503 - lr: 1.2500e-04
Epoch 35/100
56/56 [==============================] - ETA: 0s - loss: 0.3875 - accuracy: 0.8672
Restoring model weights from the end of the best epoch: 25.

Epoch 35: val_accuracy did not improve from 0.58837
56/56 [==============================] - 12s 198ms/step - loss: 0.3875 - accuracy:
0.8672 - val_loss: 1.4237 - val_accuracy: 0.5772 - lr: 1.2500e-04
Epoch 35: early stopping
```

In [90]:

```
print(f"Early Stopping at Epoch : {es_cb.best_epoch + 1}")
print(f"Train Acc at Epoch {es_cb.best_epoch} : {history.history['accuracy'][es_cb
print(f"Valid Acc at Epoch {es_cb.best_epoch} : {history.history['val_accuracy'][es
```

```
Early Stopping at Epoch : 25
Train Acc at Epoch 24 :  0.75
Valid Acc at Epoch 24 :  0.59
```

Visualizing training results

In [91]:

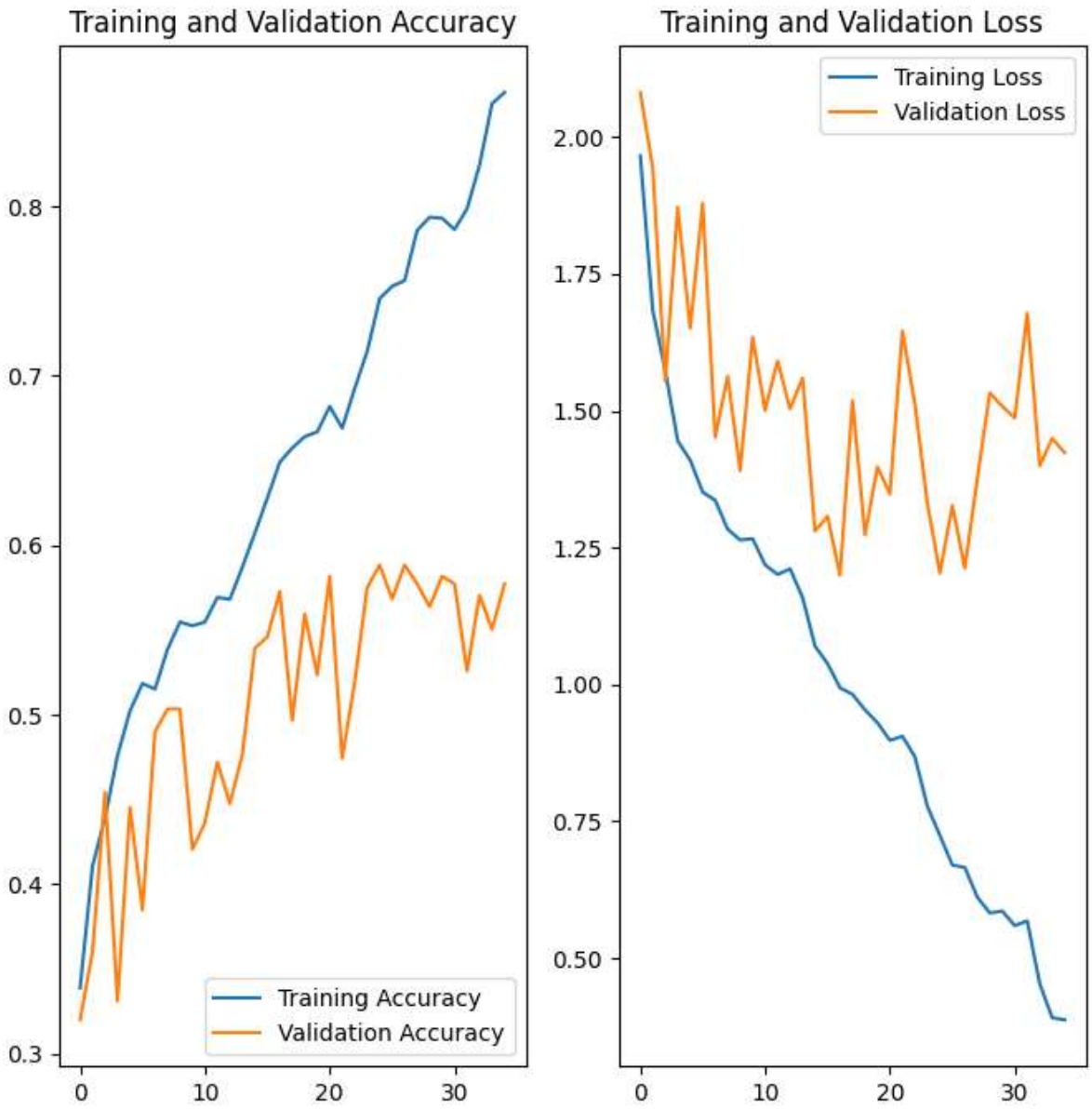
```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(len(acc))

plt.figure(figsize=(11, 4))
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()
```



Our Model early stopped at Epoch 25 and from the graph above we can analyze that after epoch 25 the model started to overfit heavily.

Baseline SqueezeNet Metrics :

- Train Acc : 75%
- Valid Acc : 59%

Confusion Matrix

In [98]:
```
val_preds = model.predict(val_ds)
val_preds.shape
```

```
14/14 [==============================] - 5s 95ms/step
```

Out[98]:
```
(447, 9)
```

In [101…
```
val_preds = np.argmax(val_preds, axis=1)
val_preds.shape
```

Out[101]:
```
(447,)
```

In [102…]
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="black" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

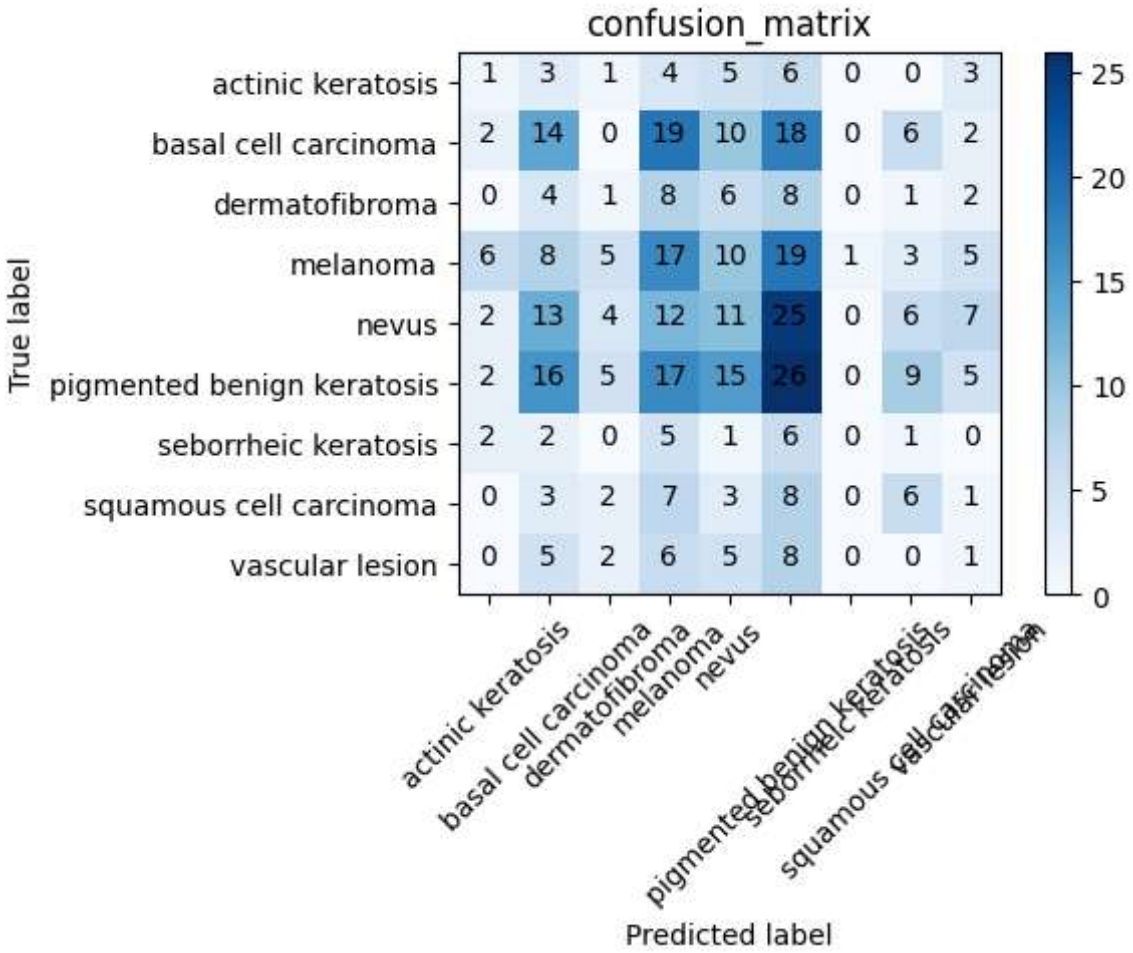
In [112…]
```
val_y = np.concatenate([y for x, y in val_ds], axis=0)
val_y.shape
```

Out[112]:
```
(447,)
```

In [115…]
```
cm_plot_labels = class_names

cm = confusion_matrix(y_true=val_y, y_pred=val_preds)
plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='confusion_matrix')
```

```
Confusion matrix, without normalization
[[ 1  3  1  4  5  6  0  0  3]
 [ 2 14  0 19 10 18  0  6  2]
 [ 0  4  1  8  6  8  0  1  2]
 [ 6  8  5 17 10 19  1  3  5]
 [ 2 13  4 12 11 25  0  6  7]
 [ 2 16  5 17 15 26  0  9  5]
 [ 2  2  0  5  1  6  0  1  0]
 [ 0  3  2  7  3  8  0  6  1]
 [ 0  5  2  6  5  8  0  0  1]]
```



From the confusion matrix we can see that the classes which contain very less samples are getting classified very incorrectly :

- 'actinic keratosis' - 1 Correct Prediction
- 'dermatofibroma' - 1 Correct Prediction
- 'seborrheic keratosis' - 0 Correct Prediction
- 'squamous cell carcinoma' - 6 Correct Prediction
- 'vascular lesion' - 1 Correct Prediction

It seems that we need to balance these classes to get a good accuracy and for the model to learn better.

Class Distribution

In [118…

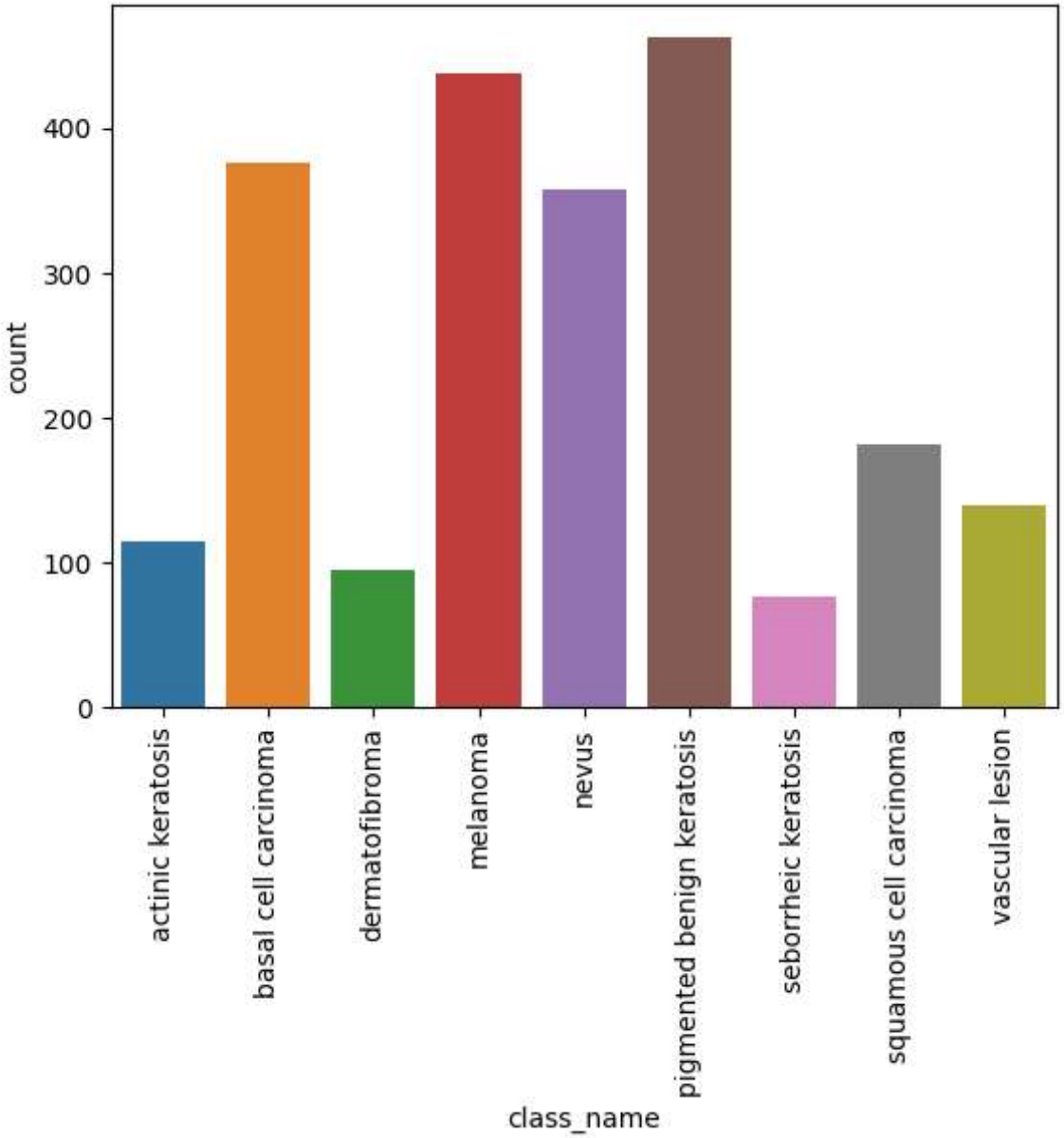
```
num_classes = len(class_names)
total = 0
all_count = []
class_name = []
for i in range(num_classes):
    count = len(list(data_dir_train.glob(class_names[i]+'/*.jpg')))
    total += count
print("total training image count = {} \n".format(total))
print("-------------------------------------")
for i in range(num_classes):
    count = len(list(data_dir_train.glob(class_names[i]+'/*.jpg')))
    print("Class name = ",class_names[i])
    print("count      = ",count)
    print("proportion = ",count/total)
```

```
    print("-------------------------------------")
    all_count.append(count)
    class_name.append(class_names[i])

temp_df = pd.DataFrame(list(zip(all_count, class_name)), columns = ['count', 'class
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()
```

```
total training image count = 2239

-------------------------------------
Class name =  actinic keratosis
count      =  114
proportion =  0.05091558731576597
-------------------------------------
Class name =  basal cell carcinoma
count      =  376
proportion =  0.16793211255024565
-------------------------------------
Class name =  dermatofibroma
count      =  95
proportion =  0.04242965609647164
-------------------------------------
Class name =  melanoma
count      =  438
proportion =  0.19562304600267977
-------------------------------------
Class name =  nevus
count      =  357
proportion =  0.15944618133095131
-------------------------------------
Class name =  pigmented benign keratosis
count      =  462
proportion =  0.20634211701652524
-------------------------------------
Class name =  seborrheic keratosis
count      =  77
proportion =  0.03439035283608754
-------------------------------------
Class name =  squamous cell carcinoma
count      =  181
proportion =  0.08083966056275123
-------------------------------------
Class name =  vascular lesion
count      =  139
proportion =  0.062081286288521664
-------------------------------------
```



- Class with least count of samples : seborrheic keratosis
- Class with highest count of samples : pigmented benign keratosis

Findings :

- We can see that the model is overfitting heavily due to high number of epochs on a static dataset.
- We'll have to put in random augmentation dynamically for each epoch and each batch so that this overfitting on training set doesn't happen and we get a robust model.
- Imbalanced Dataset : Apart four classes [carcinoma, melanoma, nevus, pigmented benign keratosis] every class has very less amount of samples i.e. imbalanced, we can actually use augmentations to correct this imbalance.
- Due to this imbalanced dataset, 5 out of our 9 classes have very less accuracy which have very less samples.

Data Augmentation

The idea here is to Augment the training dataset so that we train the model for longer without overfitting on the train set and get a good and robust model.

In [276…
```
aug_layers = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.5),
    layers.experimental.preprocessing.RandomContrast(0.3),
    layers.experimental.preprocessing.RandomZoom(0.2),
])
```

In [277…
```
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(img_height, img_width),
    layers.experimental.preprocessing.Rescaling(scale=1./255),
])
```

In [278…
```
batch_size = 32

def prepare(ds, shuffle=False, augment=False):
    ds = ds.map(lambda x, y: (resize_and_rescale(x), y))

    if shuffle:
        ds = ds.shuffle(1000)

    ds = ds.batch(batch_size)

    # Use data augmentation only on the training set.
    if augment:
        ds = ds.map(lambda x, y: (aug_layers(x, training=True), y))

    return ds
```

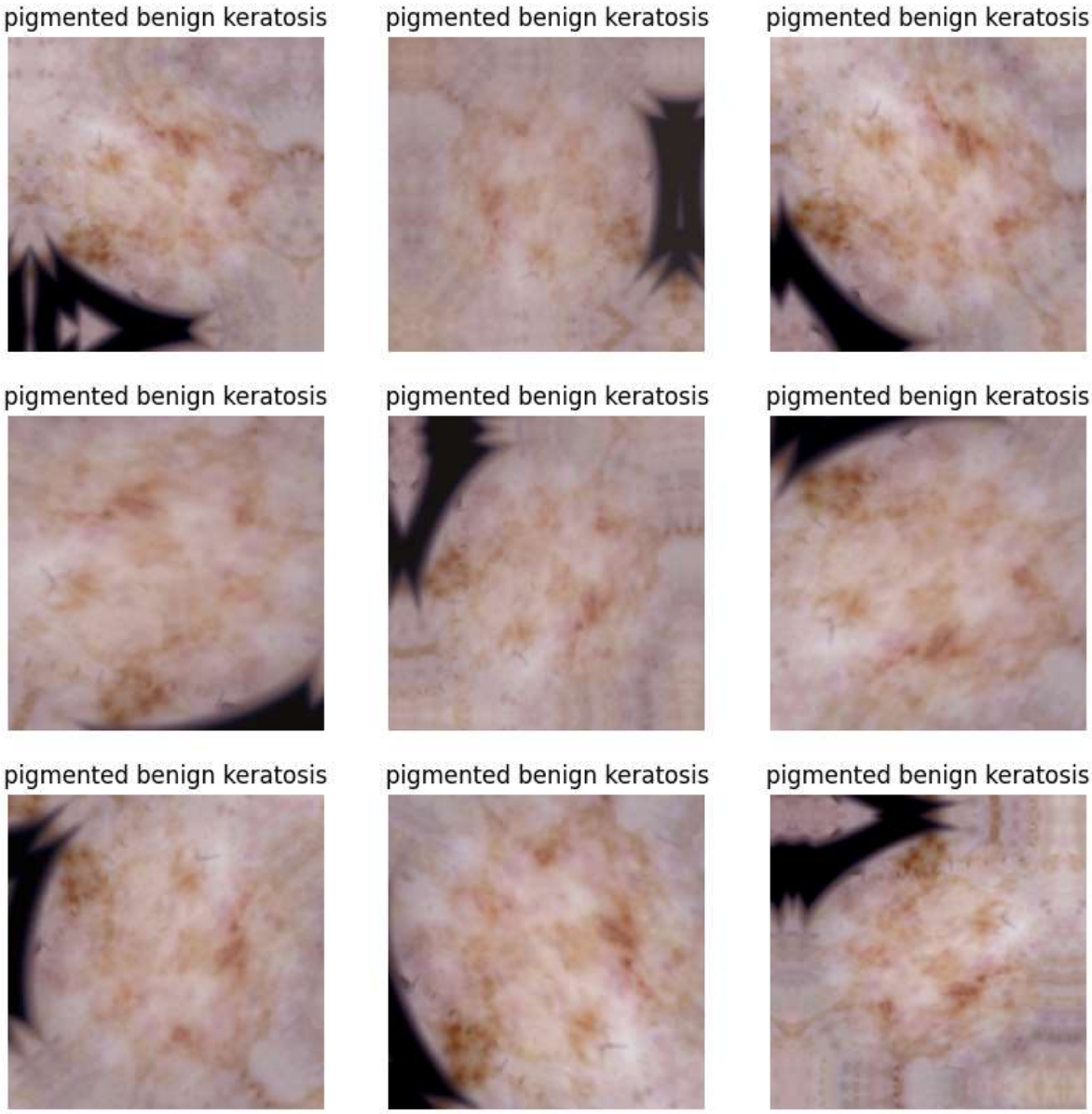
In [292…
```
train_aug_ds = prepare_dataset(train_ds, shuffle=True, augment=True)
valid_aug_ds = prepare_dataset(val_ds, shuffle=False, augment=False)
```

Looking at augmented images

In [293…
```
plt.figure(figsize=(10, 10))

images, labels = next(iter(train_aug_ds))

img, lab = images[0], labels[0].numpy()
img = tf.cast(tf.expand_dims(img, 0), tf.float32)
for index in range(9):
    ax = plt.subplot(3, 3, index + 1)
    aug_img = aug_layers(img, training=True)
    plt.imshow(aug_img[0])
    plt.title(class_names[lab])
    plt.axis("off")
```



In [295…
```
snet = SqueezeNet()

model = snet.build_model(n_classes=9, choice='bypass', input_shape=(180, 180, 3))
opt = tf.keras.optimizers.Adam(learning_rate=0.001) #
model.compile(optimizer= opt,
              loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Fals
              metrics=['accuracy'])
```

In [296…
```
ckpt_cb = tf.keras.callbacks.ModelCheckpoint(filepath="baseline_snet_weights_augmer
```

In [297…
```
epochs = 40
history = model.fit(
    train_aug_ds,
    validation_data=valid_aug_ds,
    epochs=epochs,
    callbacks=[es_cb, rdr_cb, ckpt_cb],
)
```

```
Epoch 1/100
56/56 [==============================] - ETA: 0s - loss: 1.9836 - accuracy: 0.3493
Epoch 1: val_accuracy improved from -inf to 0.25056, saving model to baseline_snet
_weights_augmented_data.hdf5
56/56 [==============================] - 32s 506ms/step - loss: 1.9836 - accuracy:
0.3493 - val_loss: 5.0896 - val_accuracy: 0.2506 - lr: 0.0010
Epoch 2/100
56/56 [==============================] - ETA: 0s - loss: 1.6846 - accuracy: 0.4062
Epoch 2: val_accuracy improved from 0.25056 to 0.34899, saving model to baseline_s
net_weights_augmented_data.hdf5
56/56 [==============================] - 28s 493ms/step - loss: 1.6846 - accuracy:
0.4062 - val_loss: 2.3297 - val_accuracy: 0.3490 - lr: 0.0010
Epoch 3/100
56/56 [==============================] - ETA: 0s - loss: 1.5987 - accuracy: 0.4353
Epoch 3: val_accuracy did not improve from 0.34899
56/56 [==============================] - 28s 486ms/step - loss: 1.5987 - accuracy:
0.4353 - val_loss: 3.5048 - val_accuracy: 0.2685 - lr: 0.0010
Epoch 4/100
56/56 [==============================] - ETA: 0s - loss: 1.4870 - accuracy: 0.4688
Epoch 4: val_accuracy improved from 0.34899 to 0.36689, saving model to baseline_s
net_weights_augmented_data.hdf5
56/56 [==============================] - 28s 492ms/step - loss: 1.4870 - accuracy:
0.4688 - val_loss: 1.8079 - val_accuracy: 0.3669 - lr: 0.0010
Epoch 5/100
56/56 [==============================] - ETA: 0s - loss: 1.4514 - accuracy: 0.4877
Epoch 5: val_accuracy did not improve from 0.36689
56/56 [==============================] - 28s 489ms/step - loss: 1.4514 - accuracy:
0.4877 - val_loss: 2.3714 - val_accuracy: 0.2662 - lr: 0.0010
Epoch 6/100
56/56 [==============================] - ETA: 0s - loss: 1.4202 - accuracy: 0.4939
Epoch 6: val_accuracy did not improve from 0.36689
56/56 [==============================] - 28s 486ms/step - loss: 1.4202 - accuracy:
0.4939 - val_loss: 2.1866 - val_accuracy: 0.3557 - lr: 0.0010
Epoch 7/100
56/56 [==============================] - ETA: 0s - loss: 1.4367 - accuracy: 0.4911
Epoch 7: val_accuracy improved from 0.36689 to 0.44295, saving model to baseline_s
net_weights_augmented_data.hdf5
56/56 [==============================] - 28s 494ms/step - loss: 1.4367 - accuracy:
0.4911 - val_loss: 1.5378 - val_accuracy: 0.4430 - lr: 0.0010
Epoch 8/100
56/56 [==============================] - ETA: 0s - loss: 1.3895 - accuracy: 0.5084
Epoch 8: val_accuracy did not improve from 0.44295
56/56 [==============================] - 28s 488ms/step - loss: 1.3895 - accuracy:
0.5084 - val_loss: 1.7348 - val_accuracy: 0.3736 - lr: 0.0010
Epoch 9/100
56/56 [==============================] - ETA: 0s - loss: 1.3845 - accuracy: 0.5095
Epoch 9: val_accuracy did not improve from 0.44295
56/56 [==============================] - 28s 488ms/step - loss: 1.3845 - accuracy:
0.5095 - val_loss: 1.9305 - val_accuracy: 0.3579 - lr: 0.0010
Epoch 10/100
56/56 [==============================] - ETA: 0s - loss: 1.3334 - accuracy: 0.5218
Epoch 10: val_accuracy did not improve from 0.44295
56/56 [==============================] - 28s 488ms/step - loss: 1.3334 - accuracy:
0.5218 - val_loss: 1.5743 - val_accuracy: 0.4273 - lr: 0.0010
Epoch 11/100
56/56 [==============================] - ETA: 0s - loss: 1.3073 - accuracy: 0.5363
Epoch 11: val_accuracy did not improve from 0.44295
56/56 [==============================] - 28s 487ms/step - loss: 1.3073 - accuracy:
0.5363 - val_loss: 1.7335 - val_accuracy: 0.3758 - lr: 0.0010
Epoch 12/100
56/56 [==============================] - ETA: 0s - loss: 1.3284 - accuracy: 0.5246
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 12: val_accuracy did not improve from 0.44295
```

```
56/56 [==============================] - 28s 489ms/step - loss: 1.3284 - accuracy:
0.5246 - val_loss: 1.8375 - val_accuracy: 0.3579 - lr: 0.0010
Epoch 13/100
56/56 [==============================] - ETA: 0s - loss: 1.2346 - accuracy: 0.5547
Epoch 13: val_accuracy improved from 0.44295 to 0.55257, saving model to baseline_
snet_weights_augmented_data.hdf5
56/56 [==============================] - 28s 494ms/step - loss: 1.2346 - accuracy:
0.5547 - val_loss: 1.2270 - val_accuracy: 0.5526 - lr: 5.0000e-04
Epoch 14/100
56/56 [==============================] - ETA: 0s - loss: 1.1972 - accuracy: 0.5686
Epoch 14: val_accuracy did not improve from 0.55257
56/56 [==============================] - 28s 493ms/step - loss: 1.1972 - accuracy:
0.5686 - val_loss: 1.6551 - val_accuracy: 0.4206 - lr: 5.0000e-04
Epoch 15/100
56/56 [==============================] - ETA: 0s - loss: 1.2374 - accuracy: 0.5670
Epoch 15: val_accuracy did not improve from 0.55257
56/56 [==============================] - 28s 493ms/step - loss: 1.2374 - accuracy:
0.5670 - val_loss: 1.5489 - val_accuracy: 0.4765 - lr: 5.0000e-04
Epoch 16/100
56/56 [==============================] - ETA: 0s - loss: 1.2059 - accuracy: 0.5681
Epoch 16: val_accuracy improved from 0.55257 to 0.57047, saving model to baseline_
snet_weights_augmented_data.hdf5
56/56 [==============================] - 28s 492ms/step - loss: 1.2059 - accuracy:
0.5681 - val_loss: 1.1949 - val_accuracy: 0.5705 - lr: 5.0000e-04
Epoch 17/100
56/56 [==============================] - ETA: 0s - loss: 1.1935 - accuracy: 0.5843
Epoch 17: val_accuracy did not improve from 0.57047
56/56 [==============================] - 28s 488ms/step - loss: 1.1935 - accuracy:
0.5843 - val_loss: 1.3210 - val_accuracy: 0.5369 - lr: 5.0000e-04
Epoch 18/100
56/56 [==============================] - ETA: 0s - loss: 1.1686 - accuracy: 0.5703
Epoch 18: val_accuracy did not improve from 0.57047
56/56 [==============================] - 28s 485ms/step - loss: 1.1686 - accuracy:
0.5703 - val_loss: 1.2783 - val_accuracy: 0.5347 - lr: 5.0000e-04
Epoch 19/100
56/56 [==============================] - ETA: 0s - loss: 1.1851 - accuracy: 0.5714
Epoch 19: val_accuracy did not improve from 0.57047
56/56 [==============================] - 28s 487ms/step - loss: 1.1851 - accuracy:
0.5714 - val_loss: 1.2792 - val_accuracy: 0.5257 - lr: 5.0000e-04
Epoch 20/100
56/56 [==============================] - ETA: 0s - loss: 1.1458 - accuracy: 0.5887
Epoch 20: val_accuracy did not improve from 0.57047
56/56 [==============================] - 28s 489ms/step - loss: 1.1458 - accuracy:
0.5887 - val_loss: 1.2985 - val_accuracy: 0.5324 - lr: 5.0000e-04
Epoch 21/100
56/56 [==============================] - ETA: 0s - loss: 1.1839 - accuracy: 0.5753
Epoch 21: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.

Epoch 21: val_accuracy did not improve from 0.57047
56/56 [==============================] - 28s 487ms/step - loss: 1.1839 - accuracy:
0.5753 - val_loss: 1.2320 - val_accuracy: 0.5526 - lr: 5.0000e-04
Epoch 22/100
56/56 [==============================] - ETA: 0s - loss: 1.0928 - accuracy: 0.6032
Epoch 22: val_accuracy improved from 0.57047 to 0.57494, saving model to baseline_
snet_weights_augmented_data.hdf5
56/56 [==============================] - 28s 491ms/step - loss: 1.0928 - accuracy:
0.6032 - val_loss: 1.2201 - val_accuracy: 0.5749 - lr: 2.5000e-04
Epoch 23/100
56/56 [==============================] - ETA: 0s - loss: 1.1022 - accuracy: 0.6027
Epoch 23: val_accuracy did not improve from 0.57494
56/56 [==============================] - 29s 496ms/step - loss: 1.1022 - accuracy:
0.6027 - val_loss: 1.1970 - val_accuracy: 0.5749 - lr: 2.5000e-04
Epoch 24/100
56/56 [==============================] - ETA: 0s - loss: 1.0786 - accuracy: 0.6194
```

```
Epoch 24: val_accuracy improved from 0.57494 to 0.57942, saving model to baseline_
snet_weights_augmented_data.hdf5
56/56 [==============================] - 29s 502ms/step - loss: 1.0786 - accuracy:
0.6194 - val_loss: 1.1560 - val_accuracy: 0.5794 - lr: 2.5000e-04
Epoch 25/100
56/56 [==============================] - ETA: 0s - loss: 1.0649 - accuracy: 0.6172
Epoch 25: val_accuracy improved from 0.57942 to 0.61969, saving model to baseline_
snet_weights_augmented_data.hdf5
56/56 [==============================] - 29s 504ms/step - loss: 1.0649 - accuracy:
0.6172 - val_loss: 1.1330 - val_accuracy: 0.6197 - lr: 2.5000e-04
Epoch 26/100
56/56 [==============================] - ETA: 0s - loss: 1.0640 - accuracy: 0.6077
Epoch 26: val_accuracy did not improve from 0.61969
56/56 [==============================] - 29s 495ms/step - loss: 1.0640 - accuracy:
0.6077 - val_loss: 1.3145 - val_accuracy: 0.5257 - lr: 2.5000e-04
Epoch 27/100
56/56 [==============================] - ETA: 0s - loss: 1.0812 - accuracy: 0.5993
Epoch 27: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 488ms/step - loss: 1.0812 - accuracy:
0.5993 - val_loss: 1.2235 - val_accuracy: 0.5749 - lr: 2.5000e-04
Epoch 28/100
56/56 [==============================] - ETA: 0s - loss: 1.0279 - accuracy: 0.6306
Epoch 28: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 485ms/step - loss: 1.0279 - accuracy:
0.6306 - val_loss: 1.1370 - val_accuracy: 0.5817 - lr: 2.5000e-04
Epoch 29/100
56/56 [==============================] - ETA: 0s - loss: 1.0594 - accuracy: 0.6166
Epoch 29: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 486ms/step - loss: 1.0594 - accuracy:
0.6166 - val_loss: 1.1628 - val_accuracy: 0.5794 - lr: 2.5000e-04
Epoch 30/100
56/56 [==============================] - ETA: 0s - loss: 1.0501 - accuracy: 0.6166
Epoch 30: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.

Epoch 30: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 488ms/step - loss: 1.0501 - accuracy:
0.6166 - val_loss: 1.1980 - val_accuracy: 0.5817 - lr: 2.5000e-04
Epoch 31/100
56/56 [==============================] - ETA: 0s - loss: 1.0116 - accuracy: 0.6217
Epoch 31: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 489ms/step - loss: 1.0116 - accuracy:
0.6217 - val_loss: 1.1448 - val_accuracy: 0.6152 - lr: 1.2500e-04
Epoch 32/100
56/56 [==============================] - ETA: 0s - loss: 0.9991 - accuracy: 0.6406
Epoch 32: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 488ms/step - loss: 0.9991 - accuracy:
0.6406 - val_loss: 1.1280 - val_accuracy: 0.5996 - lr: 1.2500e-04
Epoch 33/100
56/56 [==============================] - ETA: 0s - loss: 0.9810 - accuracy: 0.6456
Epoch 33: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 486ms/step - loss: 0.9810 - accuracy:
0.6456 - val_loss: 1.1345 - val_accuracy: 0.5906 - lr: 1.2500e-04
Epoch 34/100
56/56 [==============================] - ETA: 0s - loss: 0.9733 - accuracy: 0.6596
Epoch 34: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 485ms/step - loss: 0.9733 - accuracy:
0.6596 - val_loss: 1.2179 - val_accuracy: 0.5817 - lr: 1.2500e-04
Epoch 35/100
56/56 [==============================] - ETA: 0s - loss: 1.0015 - accuracy: 0.6434
Restoring model weights from the end of the best epoch: 25.

Epoch 35: val_accuracy did not improve from 0.61969
56/56 [==============================] - 28s 487ms/step - loss: 1.0015 - accuracy:
```

```
0.6434 - val_loss: 1.1550 - val_accuracy: 0.5794 - lr: 1.2500e-04
Epoch 35: early stopping
```

In [299…]
```
print(f"Early Stopping at Epoch : {es_cb.best_epoch + 1}")
print(f"Train Acc at Epoch {es_cb.best_epoch} : {history.history['accuracy'][es_cb
print(f"Valid Acc at Epoch {es_cb.best_epoch} : {history.history['val_accuracy'][es
```

```
Early Stopping at Epoch : 25
Train Acc at Epoch 24 :  0.62
Valid Acc at Epoch 24 :  0.62
```

Visualizing the training results after augmentation

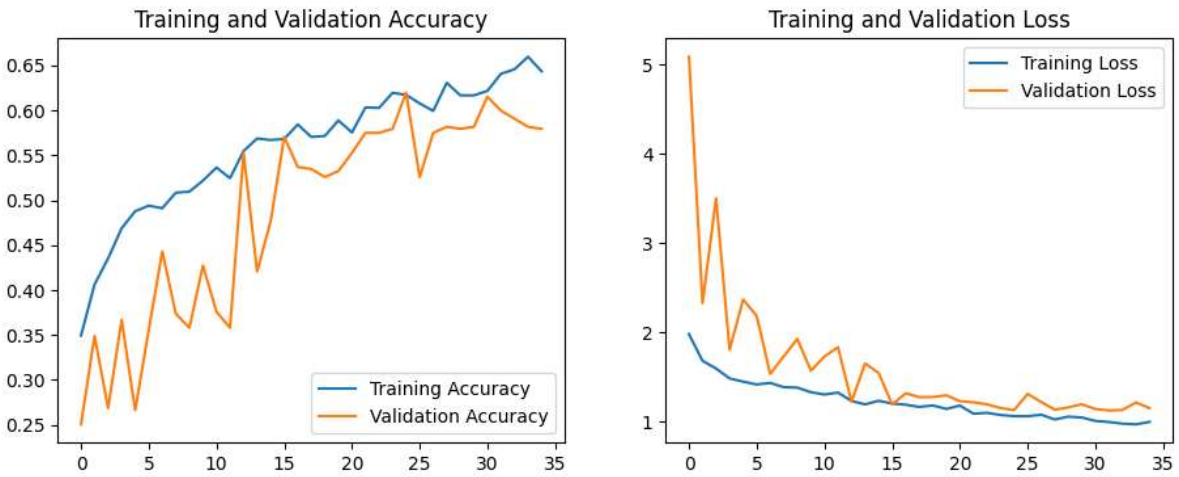
In [306…]
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(len(acc))

plt.figure(figsize=(11, 4))
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()
```



Our Model early stopped at Epoch 25 and from the graph above we can analyze that after epoch 25 the model started to overfit heavily.

Metrics after Data Augmentation :

- Train Acc : 62%
- Valid Acc : 62%

Findings :

- The training is smoother, i.e. the difference between the training curve and validation curve is smaller showing a good training.
- We can see that the validation accuracy has increased from 59 to 62, this is due to better training happening due to the augmentation layers being used.
- Training Accuracy is decreased from 75 to 62 which is a steep decrease, so there's an opportunity to add a FCN layer to fit better as this might be a underfit.

Tackling Class Imbalance

To use `Augmentor`, the following general procedure is followed:

1. Instantiate a `Pipeline` object pointing to a directory containing your initial image data set.
2. Define a number of operations to perform on this data set using your `Pipeline` object.
3. Execute these operations by calling the `Pipeline's` `sample()` method.

In [310…
```
path_to_training_dataset = "../../data/CNN_assignment/Train/"
```

In [311…
```
import Augmentor

for i in class_names:
    p = Augmentor.Pipeline(path_to_training_dataset + i)
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(500) ## We are adding 500 samples per class to make sure that none of
```

```
Initialised with 114 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/actinic keratosis\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x2037C630EB0>: 100%|██
███████| 500/500 [00:03<00:00, 163.11 Samples/s]
Initialised with 376 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/basal cell carcinoma\outpu
t.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x203
7F9D1580>: 100%|████████| 500/500 [00:03<00:00, 156.81 Samples/s]
Initialised with 95 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/dermatofibroma\output.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x203
8184B100>: 100%|████████| 500/500 [00:03<00:00, 156.46 Samples/s]
Initialised with 438 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/melanoma\output.
Processing <PIL.Image.Image image mode=RGB size=963x629 at 0x2037C611280>: 100%|██
███████| 500/500 [00:14<00:00, 34.71 Samples/s]
Initialised with 357 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/nevus\output.
Processing <PIL.Image.Image image mode=RGB size=2048x1536 at 0x2037C4A3280>: 100%|
████████| 500/500 [00:13<00:00, 38.43 Samples/s]
Initialised with 462 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/pigmented benign keratosis
\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x2037C54AF40>: 100%|██
███████| 500/500 [00:03<00:00, 157.53 Samples/s]
Initialised with 77 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/seborrheic keratosis\outpu
t.
```

```
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x2037F9AE8E0>: 100%|
████████| 500/500 [00:06<00:00, 72.13 Samples/s]
Initialised with 181 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/squamous cell carcinoma\ou
tput.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x2037C611A60>: 100%|
████████| 500/500 [00:02<00:00, 188.96 Samples/s]
Initialised with 139 image(s) found.
Output directory set to ../../data/CNN_assignment/Train/vascular lesion\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20380C140A0>: 100%|
████████| 500/500 [00:02<00:00, 184.23 Samples/s]
```

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.

In [312…
```
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg')))
print(image_count_train)
```

```
4500
```

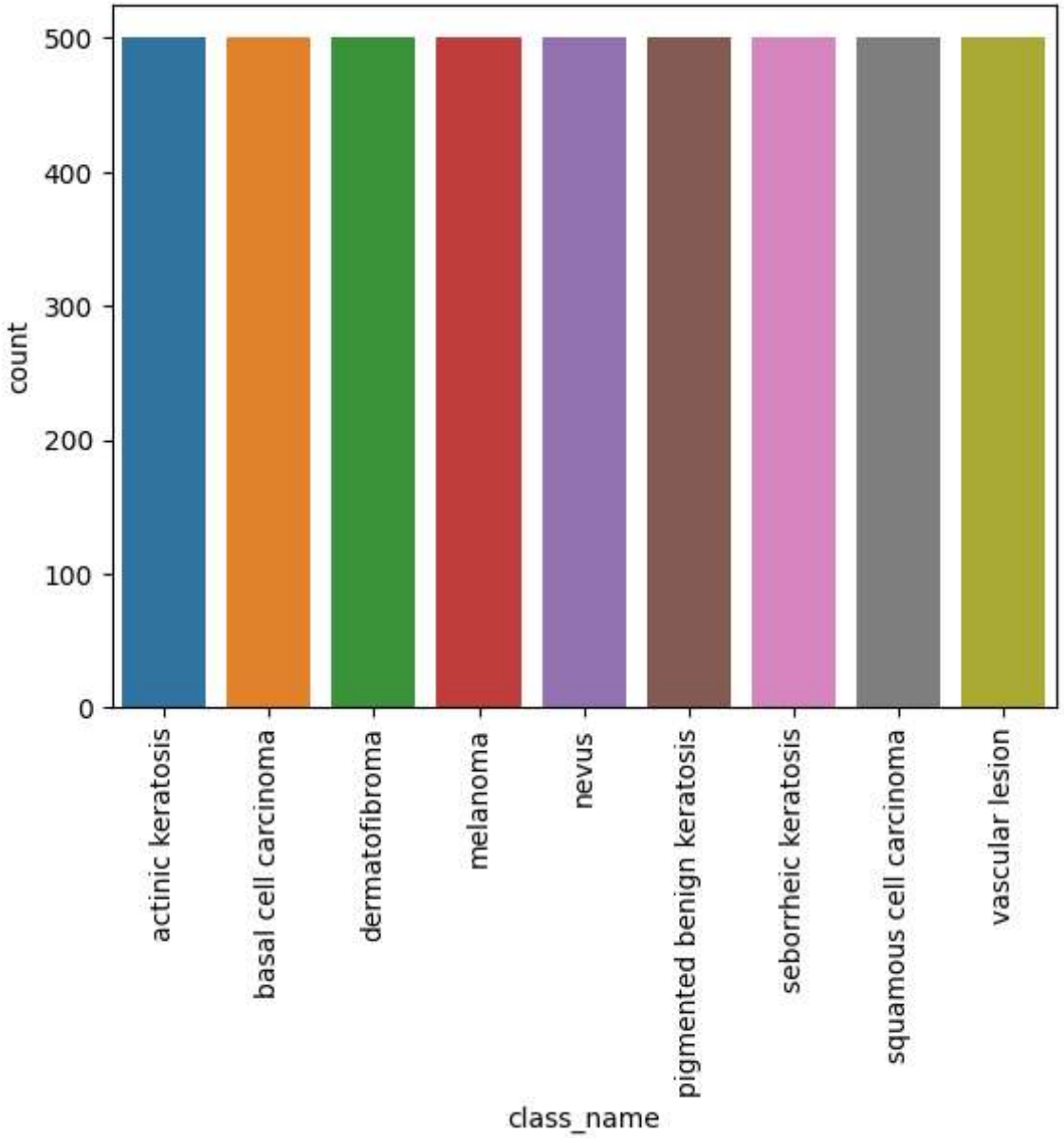
Lets see the distribution of augmented data after adding new images to the original training data.

In [333…
```
num_classes = len(class_names)
total = 0
all_count = []
class_name = []
for i in range(num_classes):
    count = len(list(data_dir_train.glob(class_names[i]+'/output/*.jpg')))
    total += count
print("total training image count = {} \n".format(total))
print("-------------------------------------")
for i in range(num_classes):
    count = len(list(data_dir_train.glob(class_names[i]+'/output/*.jpg')))
    print("Class name = ",class_names[i])
    print("count      = ",count)
    print("proportion = ",count/total)
    print("-------------------------------------")
    all_count.append(count)
    class_name.append(class_names[i])

temp_df = pd.DataFrame(list(zip(all_count, class_name)), columns = ['count', 'class
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()
```

```
total training image count = 4500

-----------------------------------
Class name =  actinic keratosis
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  basal cell carcinoma
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  dermatofibroma
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  melanoma
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  nevus
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  pigmented benign keratosis
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  seborrheic keratosis
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  squamous cell carcinoma
count      =  500
proportion =  0.1111111111111111
-----------------------------------
Class name =  vascular lesion
count      =  500
proportion =  0.1111111111111111
-----------------------------------
```



So, now we have added 500 images to all the classes to maintain some class balance. We can add more images as we want to improve training process.

Training with Balanced Dataset

In [332…
```
batch_size = 32
img_height = 180
img_width = 180
```

Creating the Training Dataset

In [338…
```
data_dir_train = pathlib.Path("../../data/CNN_assignment/Augmented Data/Train")
```

In [340…
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    seed=DEFAULT_RANDOM_SEED,
    validation_split = 0.2,
    subset = "training",
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

```
Found 4500 files belonging to 9 classes.
Using 3600 files for training.
```

Todo: Create a validation dataset

In [341…
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    seed=DEFAULT_RANDOM_SEED,
    validation_split = 0.2,
    subset = "validation",
    image_size=(img_height, img_width),
    batch_size=batch_size
)
```

```
Found 4500 files belonging to 9 classes.
Using 900 files for validation.
```

In [359…
```
snet = SqueezeNet()

model = snet.build_model(n_classes=9, choice='bypass', input_shape=(180, 180, 3))
opt = tf.keras.optimizers.Adam(learning_rate=0.001) #
model.compile(optimizer= opt,
              loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Fals
              metrics=['accuracy'])
```

In [360…
```
ckpt_cb = tf.keras.callbacks.ModelCheckpoint(
    filepath="baseline_snet_weights_augmented_and_balanced_data.hdf5", monitor="va
    verbose=1, save_best_only=True, save_weights_only=False)
```

In [361…
```
epochs = 60
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs,
    callbacks=[es_cb, rdr_cb, ckpt_cb],
)
```

```
Epoch 1/60
113/113 [==============================] - ETA: 0s - loss: 1.9229 - accuracy: 0.29
78
Epoch 1: val_accuracy improved from -inf to 0.24000, saving model to baseline_snet
_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 25s 193ms/step - loss: 1.9229 - accurac
y: 0.2978 - val_loss: 2.5652 - val_accuracy: 0.2400 - lr: 0.0010
Epoch 2/60
113/113 [==============================] - ETA: 0s - loss: 1.5878 - accuracy: 0.40
56
Epoch 2: val_accuracy improved from 0.24000 to 0.34556, saving model to baseline_s
net_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 21s 188ms/step - loss: 1.5878 - accurac
y: 0.4056 - val_loss: 1.7452 - val_accuracy: 0.3456 - lr: 0.0010
Epoch 3/60
113/113 [==============================] - ETA: 0s - loss: 1.4905 - accuracy: 0.43
78
Epoch 3: val_accuracy improved from 0.34556 to 0.37333, saving model to baseline_s
net_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 1.4905 - accurac
y: 0.4378 - val_loss: 1.9143 - val_accuracy: 0.3733 - lr: 0.0010
Epoch 4/60
113/113 [==============================] - ETA: 0s - loss: 1.3913 - accuracy: 0.47
31
Epoch 4: val_accuracy did not improve from 0.37333
113/113 [==============================] - 21s 185ms/step - loss: 1.3913 - accurac
y: 0.4731 - val_loss: 2.0217 - val_accuracy: 0.3167 - lr: 0.0010
Epoch 5/60
113/113 [==============================] - ETA: 0s - loss: 1.3174 - accuracy: 0.49
81
Epoch 5: val_accuracy improved from 0.37333 to 0.39889, saving model to baseline_s
net_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 1.3174 - accurac
y: 0.4981 - val_loss: 2.0790 - val_accuracy: 0.3989 - lr: 0.0010
Epoch 6/60
113/113 [==============================] - ETA: 0s - loss: 1.2469 - accuracy: 0.53
94
Epoch 6: val_accuracy improved from 0.39889 to 0.42778, saving model to baseline_s
net_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 1.2469 - accurac
y: 0.5394 - val_loss: 1.6360 - val_accuracy: 0.4278 - lr: 0.0010
Epoch 7/60
113/113 [==============================] - ETA: 0s - loss: 1.2123 - accuracy: 0.54
58
Epoch 7: val_accuracy did not improve from 0.42778
113/113 [==============================] - 21s 186ms/step - loss: 1.2123 - accurac
y: 0.5458 - val_loss: 2.2584 - val_accuracy: 0.2578 - lr: 0.0010
Epoch 8/60
113/113 [==============================] - ETA: 0s - loss: 1.1524 - accuracy: 0.57
36
Epoch 8: val_accuracy improved from 0.42778 to 0.49778, saving model to baseline_s
net_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 1.1524 - accurac
y: 0.5736 - val_loss: 1.3675 - val_accuracy: 0.4978 - lr: 0.0010
Epoch 9/60
113/113 [==============================] - ETA: 0s - loss: 1.0965 - accuracy: 0.58
86
Epoch 9: val_accuracy did not improve from 0.49778
113/113 [==============================] - 21s 185ms/step - loss: 1.0965 - accurac
y: 0.5886 - val_loss: 1.6300 - val_accuracy: 0.4267 - lr: 0.0010
Epoch 10/60
113/113 [==============================] - ETA: 0s - loss: 1.0288 - accuracy: 0.62
44
Epoch 10: val_accuracy improved from 0.49778 to 0.56444, saving model to baseline_
```

```
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 1.0288 - accurac
y: 0.6244 - val_loss: 1.2239 - val_accuracy: 0.5644 - lr: 0.0010
Epoch 11/60
113/113 [==============================] - ETA: 0s - loss: 0.9759 - accuracy: 0.63
36
Epoch 11: val_accuracy improved from 0.56444 to 0.64000, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.9759 - accurac
y: 0.6336 - val_loss: 1.0195 - val_accuracy: 0.6400 - lr: 0.0010
Epoch 12/60
113/113 [==============================] - ETA: 0s - loss: 0.9259 - accuracy: 0.64
94
Epoch 12: val_accuracy did not improve from 0.64000
113/113 [==============================] - 21s 185ms/step - loss: 0.9259 - accurac
y: 0.6494 - val_loss: 1.3987 - val_accuracy: 0.5289 - lr: 0.0010
Epoch 13/60
113/113 [==============================] - ETA: 0s - loss: 0.8785 - accuracy: 0.66
81
Epoch 13: val_accuracy did not improve from 0.64000
113/113 [==============================] - 21s 185ms/step - loss: 0.8785 - accurac
y: 0.6681 - val_loss: 2.8214 - val_accuracy: 0.3733 - lr: 0.0010
Epoch 14/60
113/113 [==============================] - ETA: 0s - loss: 0.8212 - accuracy: 0.69
89
Epoch 14: val_accuracy did not improve from 0.64000
113/113 [==============================] - 21s 185ms/step - loss: 0.8212 - accurac
y: 0.6989 - val_loss: 1.1867 - val_accuracy: 0.5667 - lr: 0.0010
Epoch 15/60
113/113 [==============================] - ETA: 0s - loss: 0.8463 - accuracy: 0.69
08
Epoch 15: val_accuracy did not improve from 0.64000
113/113 [==============================] - 21s 185ms/step - loss: 0.8463 - accurac
y: 0.6908 - val_loss: 1.8502 - val_accuracy: 0.4711 - lr: 0.0010
Epoch 16/60
113/113 [==============================] - ETA: 0s - loss: 0.7463 - accuracy: 0.72
39
Epoch 16: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.

Epoch 16: val_accuracy did not improve from 0.64000
113/113 [==============================] - 21s 185ms/step - loss: 0.7463 - accurac
y: 0.7239 - val_loss: 1.3682 - val_accuracy: 0.5667 - lr: 0.0010
Epoch 17/60
113/113 [==============================] - ETA: 0s - loss: 0.6094 - accuracy: 0.76
67
Epoch 17: val_accuracy improved from 0.64000 to 0.64333, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 0.6094 - accurac
y: 0.7667 - val_loss: 1.0072 - val_accuracy: 0.6433 - lr: 5.0000e-04
Epoch 18/60
113/113 [==============================] - ETA: 0s - loss: 0.5404 - accuracy: 0.80
19
Epoch 18: val_accuracy did not improve from 0.64333
113/113 [==============================] - 21s 185ms/step - loss: 0.5404 - accurac
y: 0.8019 - val_loss: 1.3050 - val_accuracy: 0.5844 - lr: 5.0000e-04
Epoch 19/60
113/113 [==============================] - ETA: 0s - loss: 0.5041 - accuracy: 0.81
28
Epoch 19: val_accuracy did not improve from 0.64333
113/113 [==============================] - 21s 185ms/step - loss: 0.5041 - accurac
y: 0.8128 - val_loss: 1.5445 - val_accuracy: 0.5422 - lr: 5.0000e-04
Epoch 20/60
113/113 [==============================] - ETA: 0s - loss: 0.4824 - accuracy: 0.81
44
```

```
Epoch 20: val_accuracy improved from 0.64333 to 0.75778, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 0.4824 - accurac
y: 0.8144 - val_loss: 0.6949 - val_accuracy: 0.7578 - lr: 5.0000e-04
Epoch 21/60
113/113 [==============================] - ETA: 0s - loss: 0.4636 - accuracy: 0.83
03
Epoch 21: val_accuracy did not improve from 0.75778
113/113 [==============================] - 21s 186ms/step - loss: 0.4636 - accurac
y: 0.8303 - val_loss: 1.2997 - val_accuracy: 0.5789 - lr: 5.0000e-04
Epoch 22/60
113/113 [==============================] - ETA: 0s - loss: 0.4240 - accuracy: 0.83
61
Epoch 22: val_accuracy did not improve from 0.75778
113/113 [==============================] - 21s 186ms/step - loss: 0.4240 - accurac
y: 0.8361 - val_loss: 0.9511 - val_accuracy: 0.6844 - lr: 5.0000e-04
Epoch 23/60
113/113 [==============================] - ETA: 0s - loss: 0.4201 - accuracy: 0.84
61
Epoch 23: val_accuracy improved from 0.75778 to 0.77333, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 190ms/step - loss: 0.4201 - accurac
y: 0.8461 - val_loss: 0.7469 - val_accuracy: 0.7733 - lr: 5.0000e-04
Epoch 24/60
113/113 [==============================] - ETA: 0s - loss: 0.3806 - accuracy: 0.85
19
Epoch 24: val_accuracy did not improve from 0.77333
113/113 [==============================] - 21s 186ms/step - loss: 0.3806 - accurac
y: 0.8519 - val_loss: 0.8438 - val_accuracy: 0.7311 - lr: 5.0000e-04
Epoch 25/60
113/113 [==============================] - ETA: 0s - loss: 0.3859 - accuracy: 0.85
94
Epoch 25: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.

Epoch 25: val_accuracy did not improve from 0.77333
113/113 [==============================] - 21s 186ms/step - loss: 0.3859 - accurac
y: 0.8594 - val_loss: 1.6877 - val_accuracy: 0.5622 - lr: 5.0000e-04
Epoch 26/60
113/113 [==============================] - ETA: 0s - loss: 0.2628 - accuracy: 0.90
56
Epoch 26: val_accuracy improved from 0.77333 to 0.80556, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.2628 - accurac
y: 0.9056 - val_loss: 0.6349 - val_accuracy: 0.8056 - lr: 2.5000e-04
Epoch 27/60
113/113 [==============================] - ETA: 0s - loss: 0.2490 - accuracy: 0.91
03
Epoch 27: val_accuracy improved from 0.80556 to 0.82111, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 190ms/step - loss: 0.2490 - accurac
y: 0.9103 - val_loss: 0.5535 - val_accuracy: 0.8211 - lr: 2.5000e-04
Epoch 28/60
113/113 [==============================] - ETA: 0s - loss: 0.2090 - accuracy: 0.92
36
Epoch 28: val_accuracy did not improve from 0.82111
113/113 [==============================] - 21s 186ms/step - loss: 0.2090 - accurac
y: 0.9236 - val_loss: 0.7166 - val_accuracy: 0.7711 - lr: 2.5000e-04
Epoch 29/60
113/113 [==============================] - ETA: 0s - loss: 0.2054 - accuracy: 0.92
44
Epoch 29: val_accuracy did not improve from 0.82111
113/113 [==============================] - 21s 186ms/step - loss: 0.2054 - accurac
y: 0.9244 - val_loss: 0.6444 - val_accuracy: 0.8100 - lr: 2.5000e-04
Epoch 30/60
```

```
113/113 [==============================] - ETA: 0s - loss: 0.2079 - accuracy: 0.92
61
Epoch 30: val_accuracy did not improve from 0.82111
113/113 [==============================] - 21s 186ms/step - loss: 0.2079 - accurac
y: 0.9261 - val_loss: 0.8458 - val_accuracy: 0.7411 - lr: 2.5000e-04
Epoch 31/60
113/113 [==============================] - ETA: 0s - loss: 0.1913 - accuracy: 0.93
08
Epoch 31: val_accuracy did not improve from 0.82111
113/113 [==============================] - 21s 186ms/step - loss: 0.1913 - accurac
y: 0.9308 - val_loss: 1.0175 - val_accuracy: 0.7267 - lr: 2.5000e-04
Epoch 32/60
113/113 [==============================] - ETA: 0s - loss: 0.1906 - accuracy: 0.92
92
Epoch 32: val_accuracy did not improve from 0.82111
113/113 [==============================] - 21s 186ms/step - loss: 0.1906 - accurac
y: 0.9292 - val_loss: 0.6264 - val_accuracy: 0.8167 - lr: 2.5000e-04
Epoch 33/60
113/113 [==============================] - ETA: 0s - loss: 0.1954 - accuracy: 0.92
78
Epoch 33: val_accuracy improved from 0.82111 to 0.82444, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.1954 - accurac
y: 0.9278 - val_loss: 0.5991 - val_accuracy: 0.8244 - lr: 2.5000e-04
Epoch 34/60
113/113 [==============================] - ETA: 0s - loss: 0.1986 - accuracy: 0.92
81
Epoch 34: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.

Epoch 34: val_accuracy did not improve from 0.82444
113/113 [==============================] - 21s 186ms/step - loss: 0.1986 - accurac
y: 0.9281 - val_loss: 0.6400 - val_accuracy: 0.8111 - lr: 2.5000e-04
Epoch 35/60
113/113 [==============================] - ETA: 0s - loss: 0.1349 - accuracy: 0.95
03
Epoch 35: val_accuracy did not improve from 0.82444
113/113 [==============================] - 21s 186ms/step - loss: 0.1349 - accurac
y: 0.9503 - val_loss: 0.6456 - val_accuracy: 0.8089 - lr: 1.2500e-04
Epoch 36/60
113/113 [==============================] - ETA: 0s - loss: 0.1302 - accuracy: 0.95
25
Epoch 36: val_accuracy did not improve from 0.82444
113/113 [==============================] - 21s 186ms/step - loss: 0.1302 - accurac
y: 0.9525 - val_loss: 0.7695 - val_accuracy: 0.7944 - lr: 1.2500e-04
Epoch 37/60
113/113 [==============================] - ETA: 0s - loss: 0.1157 - accuracy: 0.96
14
Epoch 37: val_accuracy improved from 0.82444 to 0.85222, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.1157 - accurac
y: 0.9614 - val_loss: 0.5256 - val_accuracy: 0.8522 - lr: 1.2500e-04
Epoch 38/60
113/113 [==============================] - ETA: 0s - loss: 0.1253 - accuracy: 0.95
58
Epoch 38: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 186ms/step - loss: 0.1253 - accurac
y: 0.9558 - val_loss: 0.5474 - val_accuracy: 0.8444 - lr: 1.2500e-04
Epoch 39/60
113/113 [==============================] - ETA: 0s - loss: 0.1089 - accuracy: 0.96
06
Epoch 39: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 186ms/step - loss: 0.1089 - accurac
y: 0.9606 - val_loss: 0.5582 - val_accuracy: 0.8511 - lr: 1.2500e-04
Epoch 40/60
```

```
113/113 [==============================] - ETA: 0s - loss: 0.1046 - accuracy: 0.96
28
Epoch 40: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 186ms/step - loss: 0.1046 - accurac
y: 0.9628 - val_loss: 0.5379 - val_accuracy: 0.8489 - lr: 1.2500e-04
Epoch 41/60
113/113 [==============================] - ETA: 0s - loss: 0.1025 - accuracy: 0.96
17
Epoch 41: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 185ms/step - loss: 0.1025 - accurac
y: 0.9617 - val_loss: 0.5878 - val_accuracy: 0.8367 - lr: 1.2500e-04
Epoch 42/60
113/113 [==============================] - ETA: 0s - loss: 0.0952 - accuracy: 0.96
58
Epoch 42: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 185ms/step - loss: 0.0952 - accurac
y: 0.9658 - val_loss: 0.9792 - val_accuracy: 0.7589 - lr: 1.2500e-04
Epoch 43/60
113/113 [==============================] - ETA: 0s - loss: 0.0963 - accuracy: 0.96
44
Epoch 43: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.

Epoch 43: val_accuracy did not improve from 0.85222
113/113 [==============================] - 21s 185ms/step - loss: 0.0963 - accurac
y: 0.9644 - val_loss: 0.6219 - val_accuracy: 0.8311 - lr: 1.2500e-04
Epoch 44/60
113/113 [==============================] - ETA: 0s - loss: 0.0930 - accuracy: 0.96
92
Epoch 44: val_accuracy improved from 0.85222 to 0.85333, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.0930 - accurac
y: 0.9692 - val_loss: 0.5674 - val_accuracy: 0.8533 - lr: 6.2500e-05
Epoch 45/60
113/113 [==============================] - ETA: 0s - loss: 0.0747 - accuracy: 0.97
19
Epoch 45: val_accuracy improved from 0.85333 to 0.85556, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 188ms/step - loss: 0.0747 - accurac
y: 0.9719 - val_loss: 0.5522 - val_accuracy: 0.8556 - lr: 6.2500e-05
Epoch 46/60
113/113 [==============================] - ETA: 0s - loss: 0.0762 - accuracy: 0.97
50
Epoch 46: val_accuracy did not improve from 0.85556
113/113 [==============================] - 21s 185ms/step - loss: 0.0762 - accurac
y: 0.9750 - val_loss: 0.6112 - val_accuracy: 0.8511 - lr: 6.2500e-05
Epoch 47/60
113/113 [==============================] - ETA: 0s - loss: 0.0723 - accuracy: 0.97
39
Epoch 47: val_accuracy did not improve from 0.85556
113/113 [==============================] - 21s 185ms/step - loss: 0.0723 - accurac
y: 0.9739 - val_loss: 0.5588 - val_accuracy: 0.8500 - lr: 6.2500e-05
Epoch 48/60
113/113 [==============================] - ETA: 0s - loss: 0.0694 - accuracy: 0.97
61
Epoch 48: val_accuracy improved from 0.85556 to 0.86000, saving model to baseline_
snet_weights_augmented_and_balanced_data.hdf5
113/113 [==============================] - 22s 189ms/step - loss: 0.0694 - accurac
y: 0.9761 - val_loss: 0.5518 - val_accuracy: 0.8600 - lr: 6.2500e-05
Epoch 49/60
113/113 [==============================] - ETA: 0s - loss: 0.0691 - accuracy: 0.97
61
Epoch 49: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 186ms/step - loss: 0.0691 - accurac
y: 0.9761 - val_loss: 0.5916 - val_accuracy: 0.8544 - lr: 6.2500e-05
```

```
Epoch 50/60
113/113 [==============================] - ETA: 0s - loss: 0.0701 - accuracy: 0.97
56
Epoch 50: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0701 - accurac
y: 0.9756 - val_loss: 0.5778 - val_accuracy: 0.8489 - lr: 6.2500e-05
Epoch 51/60
113/113 [==============================] - ETA: 0s - loss: 0.0759 - accuracy: 0.97
11
Epoch 51: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0759 - accurac
y: 0.9711 - val_loss: 0.5845 - val_accuracy: 0.8511 - lr: 6.2500e-05
Epoch 52/60
113/113 [==============================] - ETA: 0s - loss: 0.0662 - accuracy: 0.97
97
Epoch 52: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.

Epoch 52: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0662 - accurac
y: 0.9797 - val_loss: 0.6036 - val_accuracy: 0.8511 - lr: 6.2500e-05
Epoch 53/60
113/113 [==============================] - ETA: 0s - loss: 0.0616 - accuracy: 0.97
69
Epoch 53: val_accuracy did not improve from 0.86000
113/113 [==============================] - 22s 188ms/step - loss: 0.0616 - accurac
y: 0.9769 - val_loss: 0.5942 - val_accuracy: 0.8444 - lr: 3.1250e-05
Epoch 54/60
113/113 [==============================] - ETA: 0s - loss: 0.0565 - accuracy: 0.97
92
Epoch 54: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0565 - accurac
y: 0.9792 - val_loss: 0.5936 - val_accuracy: 0.8533 - lr: 3.1250e-05
Epoch 55/60
113/113 [==============================] - ETA: 0s - loss: 0.0516 - accuracy: 0.98
17
Epoch 55: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 186ms/step - loss: 0.0516 - accurac
y: 0.9817 - val_loss: 0.6160 - val_accuracy: 0.8456 - lr: 3.1250e-05
Epoch 56/60
113/113 [==============================] - ETA: 0s - loss: 0.0617 - accuracy: 0.97
69
Epoch 56: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0617 - accurac
y: 0.9769 - val_loss: 0.5755 - val_accuracy: 0.8544 - lr: 3.1250e-05
Epoch 57/60
113/113 [==============================] - ETA: 0s - loss: 0.0557 - accuracy: 0.98
03
Epoch 57: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 186ms/step - loss: 0.0557 - accurac
y: 0.9803 - val_loss: 0.5854 - val_accuracy: 0.8556 - lr: 3.1250e-05
Epoch 58/60
113/113 [==============================] - ETA: 0s - loss: 0.0565 - accuracy: 0.97
86Restoring model weights from the end of the best epoch: 48.

Epoch 58: val_accuracy did not improve from 0.86000
113/113 [==============================] - 21s 185ms/step - loss: 0.0565 - accurac
y: 0.9786 - val_loss: 0.5954 - val_accuracy: 0.8544 - lr: 3.1250e-05
Epoch 58: early stopping
```

In [368…

```
print(f"Early Stopping at Epoch : {es_cb.best_epoch + 1}")
print(f"Train Acc at Epoch {es_cb.best_epoch} : {history.history['accuracy'][es_cb
print(f"Valid Acc at Epoch {es_cb.best_epoch} : {history.history['val_accuracy'][es
```

```
Early Stopping at Epoch : 48
Train Acc at Epoch 47 :  0.98
Valid Acc at Epoch 47 :  0.86
```

Visualizing the model results

In [369…

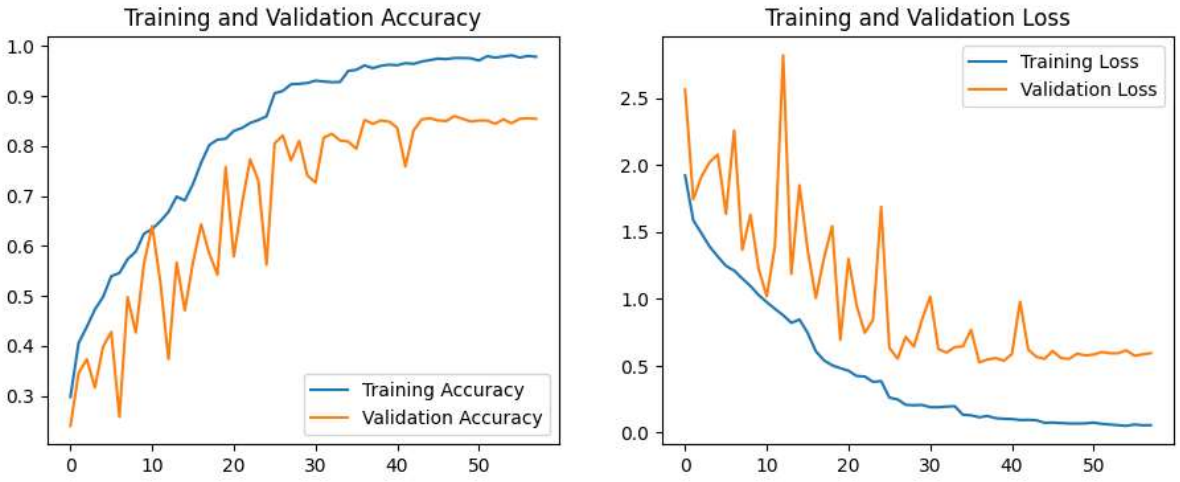
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(es_cb.best_epoch + 1)

plt.figure(figsize=(11, 4))
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()
```



Our Model early stopped at Epoch 48 and from the graph above we can analyze that after epoch 48 the training accuracy curve and validation curve has horizontally saturated and is not majorly increasing.

Metrics after Data Augmentation :

- Train Acc : **98%**
- Valid Acc : **86%**

Confusion Matrix

In [370…

```
val_preds = model.predict(val_ds)
val_preds = np.argmax(val_preds, axis=1)
val_preds.shape
```

```
29/29 [==============================] - 3s 67ms/step
```

Out[370]:
```
(900,)
```

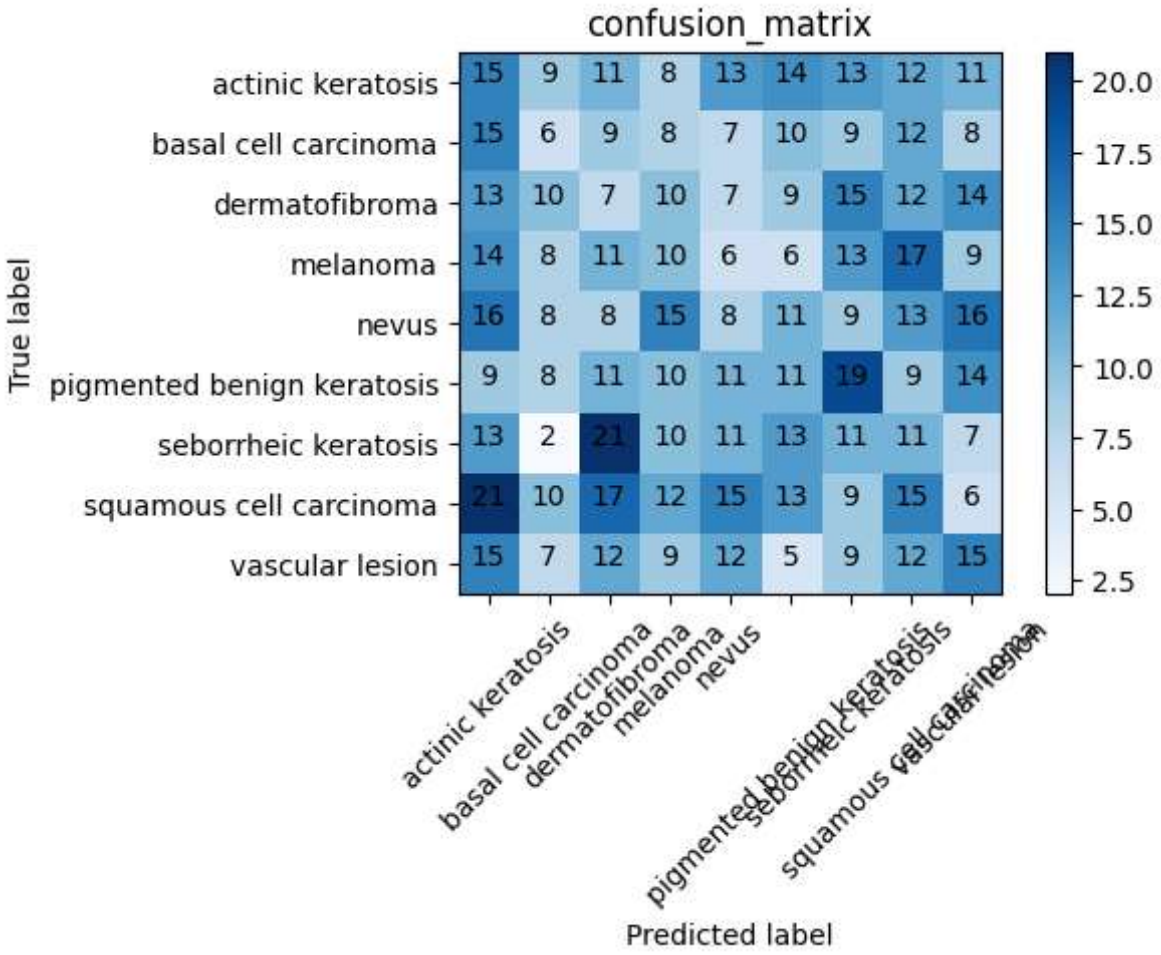
In [371…
```
val_y = np.concatenate([y for x, y in val_ds], axis=0)
val_y.shape
```

Out[371]:
```
(900,)
```

In [372…
```
cm_plot_labels = class_names

cm = confusion_matrix(y_true=val_y, y_pred=val_preds)
plot_confusion_matrix(cm=cm, classes=cm_plot_labels, title='confusion_matrix')
```

```
Confusion matrix, without normalization
[[15  9 11  8 13 14 13 12 11]
 [15  6  9  8  7 10  9 12  8]
 [13 10  7 10  7  9 15 12 14]
 [14  8 11 10  6  6 13 17  9]
 [16  8  8 15  8 11  9 13 16]
 [ 9  8 11 10 11 11 19  9 14]
 [13  2 21 10 11 13 11 11  7]
 [21 10 17 12 15 13  9 15  6]
 [15  7 12  9 12  5  9 12 15]]
```



Findings :

- The training is smoother, i.e. the difference between the training curve and validation curve is smaller showing a good training.
- Validation accuracy has increased from 62 to 86, this is due to better training happening due to the class balance done by our Augmentation Process.

- Training Accuracy has increased from 62 to 98 which is a steep increase, so there's no opportunity to fit more and we've got a robust model with 86% accuracy on validation dataset.
- I've used epochs till 50-60 as I've used a modified version of SqueezeNet and as its a compact model with no FCN, so getting a robust model would take much more epochs, hence the reason for training for more epochs.

Conclusion

- SqueezeNet is a very small and fun model to play around with!
- Score Comparision : | **Dataset Processing** | **Train Accuracy** | **Valid Accuracy** | **Epochs** | **Verdict** | |---|---|---|---|---| | **Actual Data** | 75% | 59% | 25 | **Overfit** as we have a very big gap in training and validation accuracies | | **Actual Data + Augmentation on Train** | 62% | 62% | 25 | **Underfit** as we have similar training and validation accuracies | | **Class Balanced Data** | **98%** | **86%** | **48** | **Robust Model** |

- Augmentations techniques greatly help in tackling class imbalance and the overfitting, so it is very important to use Augmentation techniques to get a robust model almost everytime!

END