In [105]:
```
import os
import pathlib
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout,BatchNorma
from keras import Sequential,Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from keras.callbacks  import ReduceLROnPlateau
from tensorflow.keras.losses import CategoricalCrossentropy

from tensorflow.keras import layers, models


from tensorflow.keras.preprocessing.image import load_img
```

Data Reading / Data Understanding

In [107]:
```
# Defining the path for both train and test images from dataset.
train_images_path = '../Org Skin cancer ISIC The International Skin Imaging Co
test_images_path = '../Org Skin cancer ISIC The International Skin Imaging Col
```

In [108]:
```
# Getting count of Images
train_images = pathlib.Path(train_images_path)
image_count_train = len(list(train_images.glob('*/*.jpg')))
print(image_count_train)
```

```
2239
```

In [110]:
```
# Getting count of Images
test_images = pathlib.Path(test_images_path)
image_count_test = len(list(test_images.glob('*/*.jpg')))
print(image_count_test)
```

```
118
```

Dataset Creation

In [111]:
```
BATCH_SIZE = 32
IMG_HEIGHT = 180
IMG_WIDTH = 180
```

In [112]:
```
# Loading the train data
# Creating train dataset from the train directory with a batch size of 32 .
# Resized images to 180*180.
```

In [112]:
```
# Loading the train data
# Creating train dataset from the train directory with a batch size of 32 .
# Resized images to 180*180.
train_ds = tf.keras.preprocessing.image_dataset_from_directory(train_images,
                                                               seed=123,
                                                               validation_spli
                                                               image_size=(IMG
                                                               batch_size=BATC
                                                               label_mode='cat
                                                               subset='trainin
```

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

In [113]:
```
## Storing Class Names in a variable for further reference
class_names = train_ds.class_names
print(class_names)
```

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

In [114]:
```
# Loading the validation data
# Creating validation dataset from the train directory with a batch size of 32
# Resized images to 180*180.
val_ds = tf.keras.preprocessing.image_dataset_from_directory(train_images,
                                                             seed=123,
                                                             validation_split=
                                                             image_size=(IMG_H
                                                             batch_size=BATCH_
                                                             label_mode='categ
                                                             subset='validatio
```

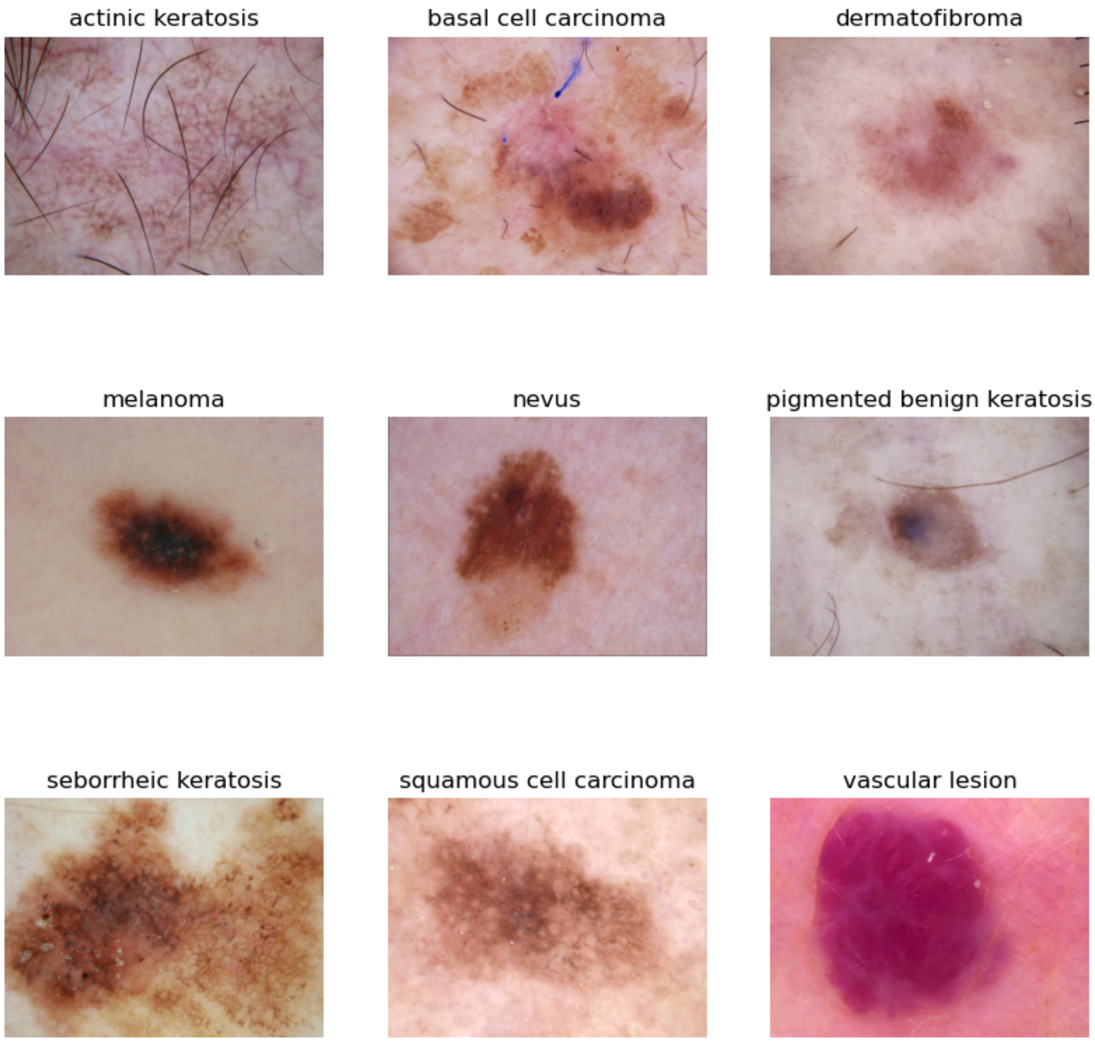
```
Found 2239 files belonging to 9 classes.
Using 447 files for validation.
```

In [115]:
```
## Visualizing one instance of all the nine classes present in the dataset

plt.figure(figsize=(10,10))
```

In [115]:
```
## Visualizing one instance of all the nine classes present in the dataset

plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    image = plt.imread(
      str(list(train_images.glob(f'{class_names[i]}/*.jpg'))[1]))
    plt.title(class_names[i])
    plt.imshow(image)
    plt.axis('off')
```



On looking the sample for each of the class , there are some classes that are visually matching a lot and it seems to be a challenging to differentiate between few of the classes for example melanoma and pigmented bening keratosis , same goes for melanoma and nevus also

Model Building & training - 1

In [116]:
```
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)
```

In [117]:
```
# 1. Building the CNN Model
model = models.Sequential([
```

In [117]:
```
# 1. Building the CNN Model
model = models.Sequential([

    # Rescaling layer to normalize pixel values
    layers.Rescaling(1./255, input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),

    # Convolutional and pooling layers
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Flatten and dense layers
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(9, activation='softmax')  # 9 classes
])
```

In [118]:
```
# Defining  optimiser and loss function
model.compile(optimizer=Adam(),
              loss=CategoricalCrossentropy(),
              metrics=['accuracy'])
```

In [119]:
```
# View the summary of all layers
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_11 (Rescaling)    (None, 180, 180, 3)       0

 conv2d_37 (Conv2D)          (None, 178, 178, 32)      896

 max_pooling2d_37 (MaxPooli  (None, 89, 89, 32)        0
 ng2D)

 conv2d_38 (Conv2D)          (None, 87, 87, 64)        18496

 max_pooling2d_38 (MaxPooli  (None, 43, 43, 64)        0
 ng2D)

 conv2d_39 (Conv2D)          (None, 41, 41, 128)       73856

 max_pooling2d_39 (MaxPooli  (None, 20, 20, 128)       0
 ng2D)

 flatten_11 (Flatten)        (None, 51200)             0

 dense_28 (Dense)            (None, 128)               6553728

 dense_29 (Dense)            (None, 9)                 1161

=================================================================
Total params: 6648137 (25.36 MB)
Trainable params: 6648137 (25.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [120]:
```
epochs = 20

history = model.fit(
```

```
Non-trainable params: 0 (0.00 Byte)
```

In [120]:
```
epochs = 20

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

```
Epoch 1/20
56/56 [==============================] - 22s 337ms/step - loss: 2.0077 - acc
uracy: 0.2411 - val_loss: 1.9441 - val_accuracy: 0.2528
```

```
Epoch 1/20
56/56 [==============================] - 22s 337ms/step - loss: 2.0077 - acc
uracy: 0.2411 - val_loss: 1.9441 - val_accuracy: 0.2528
Epoch 2/20
56/56 [==============================] - 19s 340ms/step - loss: 1.6820 - acc
uracy: 0.3945 - val_loss: 1.5009 - val_accuracy: 0.4922
Epoch 3/20
56/56 [==============================] - 18s 330ms/step - loss: 1.5521 - acc
uracy: 0.4559 - val_loss: 1.4665 - val_accuracy: 0.5034
Epoch 4/20
56/56 [==============================] - 21s 367ms/step - loss: 1.3960 - acc
uracy: 0.5089 - val_loss: 1.4706 - val_accuracy: 0.5011
Epoch 5/20
56/56 [==============================] - 22s 392ms/step - loss: 1.3445 - acc
uracy: 0.5379 - val_loss: 1.4451 - val_accuracy: 0.5011
Epoch 6/20
56/56 [==============================] - 22s 396ms/step - loss: 1.2476 - acc
uracy: 0.5502 - val_loss: 1.3908 - val_accuracy: 0.5056
Epoch 7/20
56/56 [==============================] - 22s 394ms/step - loss: 1.1692 - acc
uracy: 0.5982 - val_loss: 1.3996 - val_accuracy: 0.4966
Epoch 8/20
56/56 [==============================] - 22s 392ms/step - loss: 1.0880 - acc
uracy: 0.6222 - val_loss: 1.4509 - val_accuracy: 0.4698
Epoch 9/20
56/56 [==============================] - 22s 396ms/step - loss: 1.0974 - acc
uracy: 0.6049 - val_loss: 1.4444 - val_accuracy: 0.5414
Epoch 10/20
56/56 [==============================] - 22s 391ms/step - loss: 1.0041 - acc
uracy: 0.6417 - val_loss: 1.5444 - val_accuracy: 0.5391
Epoch 11/20
56/56 [==============================] - 23s 413ms/step - loss: 0.9363 - acc
uracy: 0.6657 - val_loss: 1.6747 - val_accuracy: 0.4855
Epoch 12/20
56/56 [==============================] - 24s 423ms/step - loss: 0.8329 - acc
uracy: 0.6987 - val_loss: 1.5147 - val_accuracy: 0.5369
Epoch 13/20
56/56 [==============================] - 24s 436ms/step - loss: 0.7574 - acc
uracy: 0.7160 - val_loss: 1.6505 - val_accuracy: 0.5391
Epoch 14/20
56/56 [==============================] - 24s 420ms/step - loss: 0.7057 - acc
uracy: 0.7511 - val_loss: 1.7491 - val_accuracy: 0.5056
Epoch 15/20
56/56 [==============================] - 24s 422ms/step - loss: 0.7164 - acc
uracy: 0.7400 - val_loss: 1.8976 - val_accuracy: 0.4899
Epoch 16/20
56/56 [==============================] - 25s 442ms/step - loss: 0.6039 - acc
uracy: 0.7863 - val_loss: 2.0724 - val_accuracy: 0.5459
Epoch 17/20
56/56 [==============================] - 25s 442ms/step - loss: 0.6107 - acc
uracy: 0.7896 - val_loss: 2.6384 - val_accuracy: 0.5190
Epoch 18/20
56/56 [==============================] - 24s 434ms/step - loss: 0.5624 - acc
uracy: 0.7924 - val_loss: 2.1275 - val_accuracy: 0.5123
Epoch 19/20
56/56 [==============================] - 26s 471ms/step - loss: 0.4716 - acc
uracy: 0.8315 - val_loss: 2.4862 - val_accuracy: 0.5101
Epoch 20/20
56/56 [==============================] - 25s 441ms/step - loss: 0.3810 - acc
uracy: 0.8616 - val_loss: 2.3023 - val_accuracy: 0.5280
```

In [121]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
```

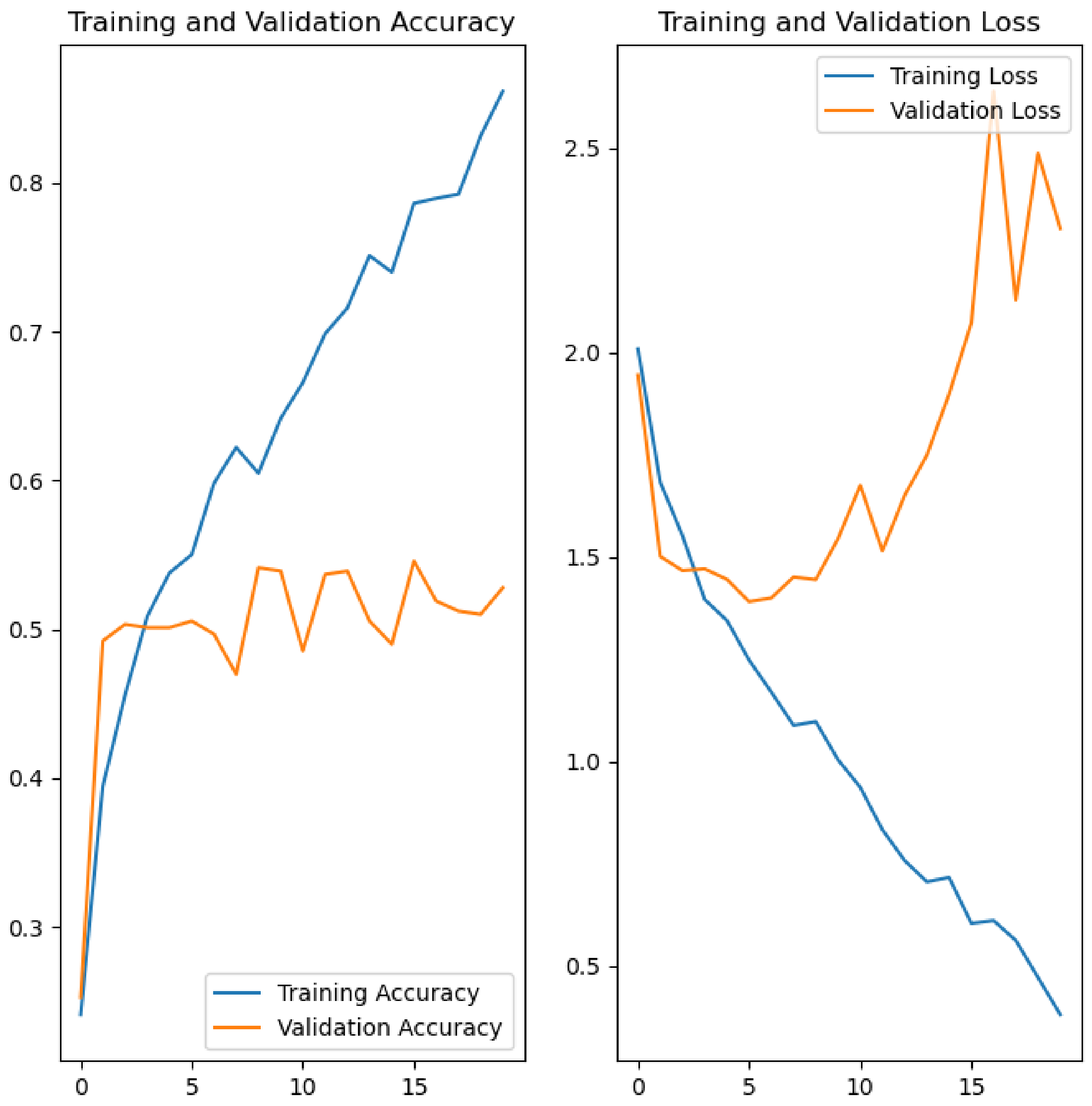
In [121]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Observations

- the model is learning, but there is room for improvement in terms of overfitting and validation performance.
- The model is overfitting as the training accuracy is significantly higher than the validation accuracy.

- the model is learning, but there is room for improvement in terms of overfitting and validation performance.
- The model is overfitting as the training accuracy is significantly higher than the validation accuracy.

Model Building & training - 2

In [ ]:
```
# Due to limited training data , the above model we created seems to overfit ,
# augmentation so that model can have training on more images that would be ge
```

In [123]:
```
# Adding data augmentation as a layer helps the model to avoid overfitting by
# artificially increasing the diversity of the training dataset
# Defining  data augmentation strategy to resolve underfitting/overfitting
data_augmentation = tf.keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical",
                                                 input_shape=(IMG_HEIGHT,
                                                              IMG_WIDTH,
                                                              3)),
    layers.experimental.preprocessing.RandomRotation(0.2),
    layers.experimental.preprocessing.RandomZoom(0.2),
    layers.experimental.preprocessing.RandomContrast(0.2)
  ]
)
```

In [124]:
```
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)
```

In [125]:
```
# 1. Building the CNN Model
model = models.Sequential([
    # Rescaling layer to normalize pixel values
    layers.Rescaling(1./255, input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    data_augmentation,
    # Convolutional and pooling layers
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Flatten and dense layers
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(9, activation='softmax')  # 9 classes
])
```

In [126]:
```
model.compile(optimizer=Adam(),
              loss=CategoricalCrossentropy(),
              metrics=['accuracy'])
```

In [127]:
```
model.summary()
```

```
Model: "sequential_14"
```

In [127]:
```
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_12 (Rescaling)    (None, 180, 180, 3)       0         
                                                                 
 sequential_13 (Sequential)  (None, 180, 180, 3)       0         
                                                                 
 conv2d_40 (Conv2D)          (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d_40 (MaxPooli  (None, 89, 89, 32)        0         
 ng2D)                                                           
                                                                 
 conv2d_41 (Conv2D)          (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_41 (MaxPooli  (None, 43, 43, 64)        0         
 ng2D)                                                           
                                                                 
 conv2d_42 (Conv2D)          (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_42 (MaxPooli  (None, 20, 20, 128)       0         
 ng2D)                                                           
                                                                 
 flatten_12 (Flatten)        (None, 51200)             0         
                                                                 
 dense_30 (Dense)            (None, 128)               6553728   
                                                                 
 dense_31 (Dense)            (None, 9)                 1161      
                                                                 
=================================================================
Total params: 6648137 (25.36 MB)
Trainable params: 6648137 (25.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [128]:
```
epochs = 20

history = model.fit(
```

In [128]:
```
epochs = 20

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

```
Epoch 1/20
56/56 [==============================] - 22s 373ms/step - loss: 2.0003 - acc
uracy: 0.2494 - val_loss: 1.7968 - val_accuracy: 0.3669
```

```
Epoch 1/20
56/56 [==============================] - 22s 373ms/step - loss: 2.0003 - acc
uracy: 0.2494 - val_loss: 1.7968 - val_accuracy: 0.3669
Epoch 2/20
56/56 [==============================] - 22s 402ms/step - loss: 1.7477 - acc
uracy: 0.3594 - val_loss: 1.5539 - val_accuracy: 0.5034
Epoch 3/20
56/56 [==============================] - 26s 462ms/step - loss: 1.5072 - acc
uracy: 0.4816 - val_loss: 1.5004 - val_accuracy: 0.4720
Epoch 4/20
56/56 [==============================] - 26s 459ms/step - loss: 1.5671 - acc
uracy: 0.4492 - val_loss: 1.4507 - val_accuracy: 0.4944
Epoch 5/20
56/56 [==============================] - 25s 452ms/step - loss: 1.4344 - acc
uracy: 0.4989 - val_loss: 1.4515 - val_accuracy: 0.4855
Epoch 6/20
56/56 [==============================] - 25s 448ms/step - loss: 1.4113 - acc
uracy: 0.5017 - val_loss: 1.4045 - val_accuracy: 0.5078
Epoch 7/20
56/56 [==============================] - 25s 448ms/step - loss: 1.3431 - acc
uracy: 0.5257 - val_loss: 1.3834 - val_accuracy: 0.4989
Epoch 8/20
56/56 [==============================] - 25s 438ms/step - loss: 1.3543 - acc
uracy: 0.5140 - val_loss: 1.3540 - val_accuracy: 0.5123
Epoch 9/20
56/56 [==============================] - 25s 441ms/step - loss: 1.3251 - acc
uracy: 0.5262 - val_loss: 1.4366 - val_accuracy: 0.5347
Epoch 10/20
56/56 [==============================] - 24s 432ms/step - loss: 1.3564 - acc
uracy: 0.5190 - val_loss: 1.3906 - val_accuracy: 0.5257
Epoch 11/20
56/56 [==============================] - 25s 448ms/step - loss: 1.3074 - acc
uracy: 0.5246 - val_loss: 1.3201 - val_accuracy: 0.5280
Epoch 12/20
56/56 [==============================] - 25s 452ms/step - loss: 1.2742 - acc
uracy: 0.5463 - val_loss: 1.3369 - val_accuracy: 0.5190
Epoch 13/20
56/56 [==============================] - 25s 452ms/step - loss: 1.2647 - acc
uracy: 0.5340 - val_loss: 1.3169 - val_accuracy: 0.5570
Epoch 14/20
56/56 [==============================] - 25s 445ms/step - loss: 1.2546 - acc
uracy: 0.5485 - val_loss: 1.3478 - val_accuracy: 0.5324
Epoch 15/20
56/56 [==============================] - 25s 440ms/step - loss: 1.2482 - acc
uracy: 0.5340 - val_loss: 1.4145 - val_accuracy: 0.5190
Epoch 16/20
56/56 [==============================] - 25s 442ms/step - loss: 1.2551 - acc
uracy: 0.5485 - val_loss: 1.2865 - val_accuracy: 0.5414
Epoch 17/20
56/56 [==============================] - 24s 435ms/step - loss: 1.2113 - acc
uracy: 0.5698 - val_loss: 1.3043 - val_accuracy: 0.5235
Epoch 18/20
56/56 [==============================] - 25s 438ms/step - loss: 1.2480 - acc
uracy: 0.5290 - val_loss: 1.3246 - val_accuracy: 0.5168
Epoch 19/20
56/56 [==============================] - 24s 428ms/step - loss: 1.2356 - acc
uracy: 0.5519 - val_loss: 1.3446 - val_accuracy: 0.5123
Epoch 20/20
56/56 [==============================] - 24s 428ms/step - loss: 1.2032 - acc
uracy: 0.5625 - val_loss: 1.2545 - val_accuracy: 0.5638
```

In [129]:

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

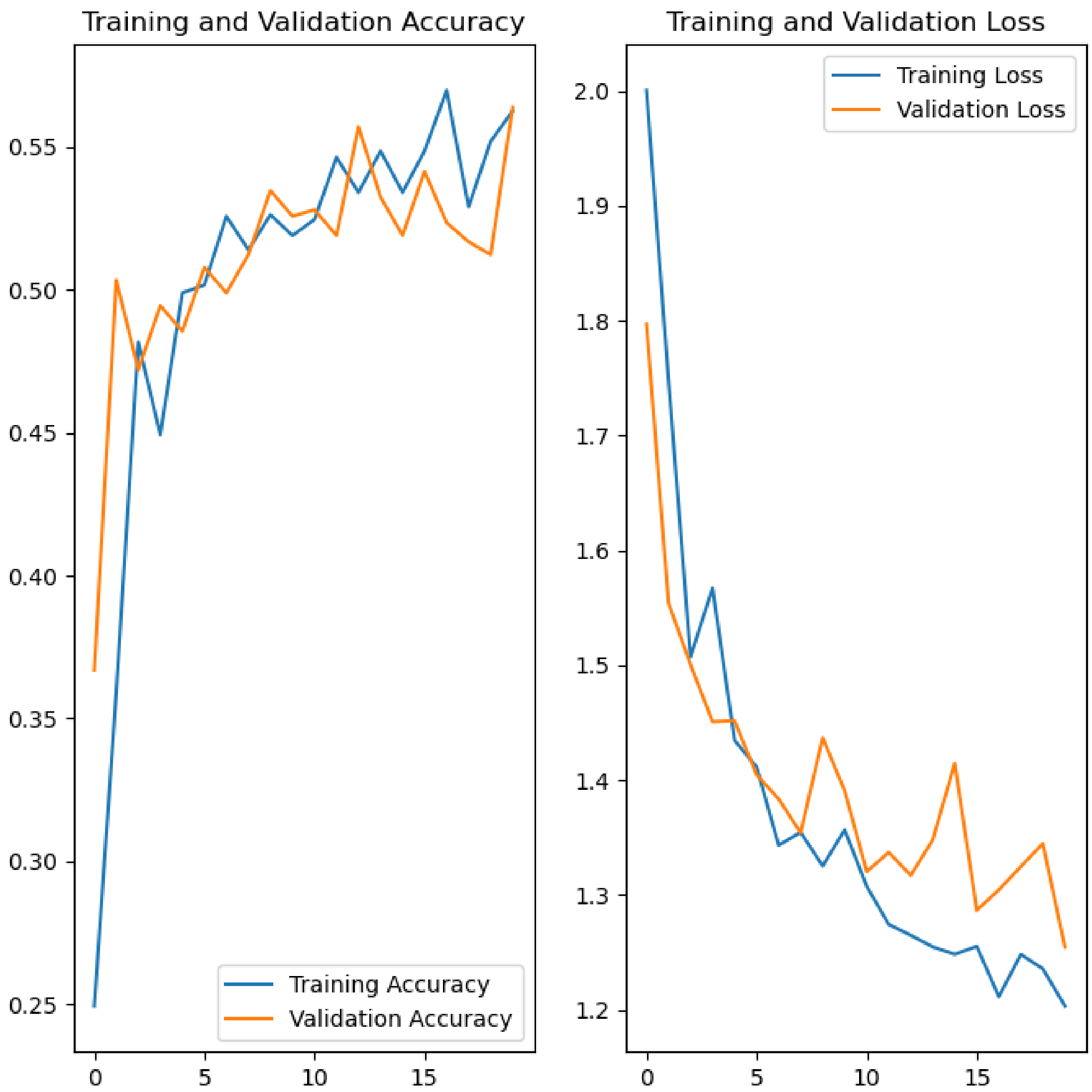
In [129]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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()
```



Observations

- With the data augmentation strategy applied , the overfitting problem seems to be fixed as how there is not much gap between training and validation dataset
- But the accuracy levels have not reached the expectation. It could be there an class imbalance as well as low data count of Images , a reason of this.

- With the data augmentation strategy applied, the overfitting problem seems to be fixed as now there is not much gap between training and validation dataset
- But the accuracy levels have not reached the expectation. It could be there an class imbalance as well as low data count of images , a reason of this.

Class Imbalance

In [131]:
```
##Find the distribution of classes in the training dataset
def class_distribution_count(directory):

    #count number of image in each classes
    count= []
    for path in pathlib.Path(directory).iterdir():
        if path.is_dir():
            count.append(len([name for name in os.listdir(path)
                               if os.path.isfile(os.path.join(path, name))]))

    #name of the classes
    sub_directory = [name for name in os.listdir(directory)
                    if os.path.isdir(os.path.join(directory, name))]

    #return dataframe with image count and class.
    return pd.DataFrame(list(zip(sub_directory,count)),columns =['Class', 'No.

df = class_distribution_count(train_images_path)
df
```

Out[131]:

	Class	No. of Image
0	actinic keratosis	114
1	basal cell carcinoma	376
2	dermatofibroma	95
3	melanoma	438
4	nevus	357
5	pigmented benign keratosis	462
6	seborrheic keratosis	77
7	squamous cell carcinoma	181
8	vascular lesion	139

Class Distribution QA

Which class has the least number of samples?
seborrheic keratosis (77)

Which classes dominate the data in terms of the proportionate number of samples?
pigmented benign keratosis (462)

In [132]:
```
train_images_path
```

Out[132]:
```
'../Org Skin cancer ISIC The International Skin Imaging Collaboration/Trai
n/'
```

In [137]:
```
train_images_path = '../Augmented Skin cancer ISIC The International Skin Imag
```

In [138]:
```
train_images = pathlib.Path(train_images_path)
test_images = pathlib.Path(test_images_path)
```

In [138]:
```
train_images = pathlib.Path(train_images_path)
test_images = pathlib.Path(test_images_path)
```

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

In [140]:
```
!pip install Augmentor
```

```
Requirement already satisfied: Augmentor in c:\anaconda\lib\site-packages
(0.2.12)
Requirement already satisfied: tqdm>=4.9.0 in c:\anaconda\lib\site-packages
(from Augmentor) (4.64.1)
Requirement already satisfied: Pillow>=5.2.0 in c:\anaconda\lib\site-package
s (from Augmentor) (9.2.0)
Requirement already satisfied: numpy>=1.11.0 in c:\anaconda\lib\site-package
s (from Augmentor) (1.24.3)
Requirement already satisfied: colorama in c:\anaconda\lib\site-packages (fr
om tqdm>=4.9.0->Augmentor) (0.4.6)
```

In [141]:
```
import Augmentor
# max_images = max([len(os.listdir(train_images / i)) for i in class_names])
# max_images = round(max_images / 100) * 100
```

In [141]:
```
import Augmentor
# max_images = max([len(os.listdir(train_images / i)) for i in class_names])
# max_images = round(max_images / 100) * 100
addCount = 500
for i in class_names:
    train_images_sub = train_images /  i
    classes_image_count_train = len(list(train_images_sub.glob('*.jpg')))
    p = Augmentor.Pipeline(train_images_sub)
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(addCount)

# On a side note , would like to add here that I initially thought to get the
# class which was coming around 463 , so Ideally to remove the class imbalance
# 100 which was coming 500. But having this set of images was failing to achie
# was coming around 50 to 60 % only , so I added 500 more images to each of th
# which means that apart of taking care of class imbalance , we should also to
# of images so that model can learn better
```

```
Initialised with 114 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\actinic keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20E1F73A8B0>: 1
00%|██████████| 500/500 [00:01<00:00, 412.36 Samples/s]

Initialised with 376 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\basal cell carcinoma\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20E205B5340>: 1
00%|██████████| 500/500 [00:01<00:00, 399.04 Samples/s]

Initialised with 95 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\dermatofibroma\output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at
0x20E1FACAA60>: 100%|██████████| 500/500 [00:01<00:00, 390.01 Samples/s]

Initialised with 438 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\melanoma\output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=3072x2304
at 0x20E20B39970>: 100%|██████████| 500/500 [00:06<00:00, 78.85 Samples/s]

Initialised with 357 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\nevus\output.

Processing <PIL.Image.Image image mode=RGB size=767x576 at 0x20E1F473F70>: 1
00%|██████████| 500/500 [00:05<00:00, 85.47 Samples/s]

Initialised with 462 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\pigmented benign keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20E1FACA880>: 1
00%|██████████| 500/500 [00:01<00:00, 381.83 Samples/s]

Initialised with 77 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\seborrheic keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x20E20973040>:
100%|██████████| 500/500 [00:02<00:00, 175.00 Samples/s]
Initialised with 188 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\squamous cell carcinoma\output.
```

```
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x20E20973040>:
100%|██████████| 500/500 [00:02<00:00, 175.00 Samples/s]
Initialised with 188 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\squamous cell carcinoma\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20E1F687C40>: 1
00%|██████████| 500/500 [00:01<00:00, 353.08 Samples/s]

Initialised with 139 image(s) found.
Output directory set to ..\Augmented Skin cancer ISIC The International Skin
Imaging Collaboration\Train\vascular lesion\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x20E1FAC51F0>: 1
00%|██████████| 500/500 [00:01<00:00, 353.21 Samples/s]
```

Model Building & training - 3

In [142]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(train_images,
                                                               seed=123,
                                                               validation_spli
                                                               image_size=(IMG
                                                               batch_size=BATC
                                                               label_mode='cat
                                                               subset='trainir
```

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

In [143]:
```
# Loading the validation data
# using seed=123 while creating dataset using tf.keras.preprocessing.image_dat
# resizing images to the size img_height*img_width, while writting the dataset
val_ds = tf.keras.preprocessing.image_dataset_from_directory(train_images,
                                                             seed=123,
                                                             validation_split=
                                                             image_size=(IMG_H
                                                             batch_size=BATCH_
                                                             label_mode='categ
                                                             subset='validatic
```

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

In [144]:
```
class_names = train_ds.class_names
```

In [145]:
```
model = tf.keras.Sequential([

    layers.experimental.preprocessing.Rescaling(1./255,input_shape=(IMG_HEIGHT
```

In [145]:
```
model = tf.keras.Sequential([

    layers.experimental.preprocessing.Rescaling(1./255,input_shape=(IMG_HEIGHT

    layers.Conv2D(32,kernel_size=(3,3),activation='relu'),
    layers.MaxPool2D(pool_size=(2,2)),

    layers.Conv2D(64,kernel_size=(3,3),activation='relu'),
    layers.MaxPool2D(pool_size=(2,2)),

    layers.Conv2D(128,kernel_size=(3,3),activation='relu'),
    layers.MaxPool2D(pool_size=(2,2)),

    layers.Conv2D(256,kernel_size=(3,3),activation='relu'),
    layers.MaxPool2D(pool_size=(2,2)),

    layers.Dropout(0.5),

    #Flatten Layer
    layers.Flatten(),

    #Dense Layer
    layers.Dense(256,activation='relu'),
    layers.Dropout(0.25),

    layers.Dense(128,activation='relu'),
    layers.Dropout(0.25),

    layers.Dense(64,activation='relu'),
    layers.Dropout(0.25),

    layers.Dense(len(class_names),activation='softmax')

])
```

In [146]:
```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [147]:
```
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.
                              patience=5, min_lr=0.0001)
```

In [148]:
```
model.summary()
```

```
Model: "sequential_15"
```

In [148]:
```
model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_13 (Rescaling)    (None, 180, 180, 3)       0

 conv2d_43 (Conv2D)          (None, 178, 178, 32)      896

 max_pooling2d_43 (MaxPooli  (None, 89, 89, 32)        0
 ng2D)

 conv2d_44 (Conv2D)          (None, 87, 87, 64)        18496

 max_pooling2d_44 (MaxPooli  (None, 43, 43, 64)        0
 ng2D)

 conv2d_45 (Conv2D)          (None, 41, 41, 128)       73856

 max_pooling2d_45 (MaxPooli  (None, 20, 20, 128)       0
 ng2D)

 conv2d_46 (Conv2D)          (None, 18, 18, 256)       295168

 max_pooling2d_46 (MaxPooli  (None, 9, 9, 256)         0
 ng2D)

 dropout_28 (Dropout)        (None, 9, 9, 256)         0

 flatten_13 (Flatten)        (None, 20736)             0

 dense_32 (Dense)            (None, 256)               5308672

 dropout_29 (Dropout)        (None, 256)               0

 dense_33 (Dense)            (None, 128)               32896

 dropout_30 (Dropout)        (None, 128)               0

 dense_34 (Dense)            (None, 64)                8256

 dropout_31 (Dropout)        (None, 64)                0

 dense_35 (Dense)            (None, 9)                 585

=================================================================
Total params: 5738825 (21.89 MB)
Trainable params: 5738825 (21.89 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [149]:
```
epochs = 30
```

In [150]:
```
history = model.fit(train_ds, validation_data=val_ds, epochs=epochs, callbacks
```

```
accuracy: 0.7311 - val_loss: 0.6263 - val_accuracy: 0.7765 - lr: 0.0010
```

In [150]:
```
history = model.fit(train_ds, validation_data=val_ds, epochs=epochs, callbacks
```

```
accuracy: 0.7311 - val_loss: 0.6263 - val_accuracy: 0.7765 - lr: 0.0010
Epoch 25/30
169/169 [==============================] - 79s 466ms/step - loss: 0.6408 -
accuracy: 0.7582 - val_loss: 0.6206 - val_accuracy: 0.7736 - lr: 0.0010
Epoch 26/30
169/169 [==============================] - 80s 475ms/step - loss: 0.6536 -
accuracy: 0.7559 - val_loss: 0.6825 - val_accuracy: 0.7803 - lr: 0.0010
Epoch 27/30
169/169 [==============================] - 80s 473ms/step - loss: 0.5932 -
accuracy: 0.7726 - val_loss: 0.6282 - val_accuracy: 0.7951 - lr: 0.0010
Epoch 28/30
169/169 [==============================] - 79s 466ms/step - loss: 0.5865 -
accuracy: 0.7810 - val_loss: 0.5701 - val_accuracy: 0.8040 - lr: 0.0010
Epoch 29/30
169/169 [==============================] - 78s 462ms/step - loss: 0.5628 -
accuracy: 0.7836 - val_loss: 0.5578 - val_accuracy: 0.8048 - lr: 0.0010
Epoch 30/30
169/169 [==============================] - 79s 466ms/step - loss: 0.5268 -
accuracy: 0.8079 - val_loss: 0.5327 - val_accuracy: 0.8196 - lr: 0.0010
```

In [151]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
```

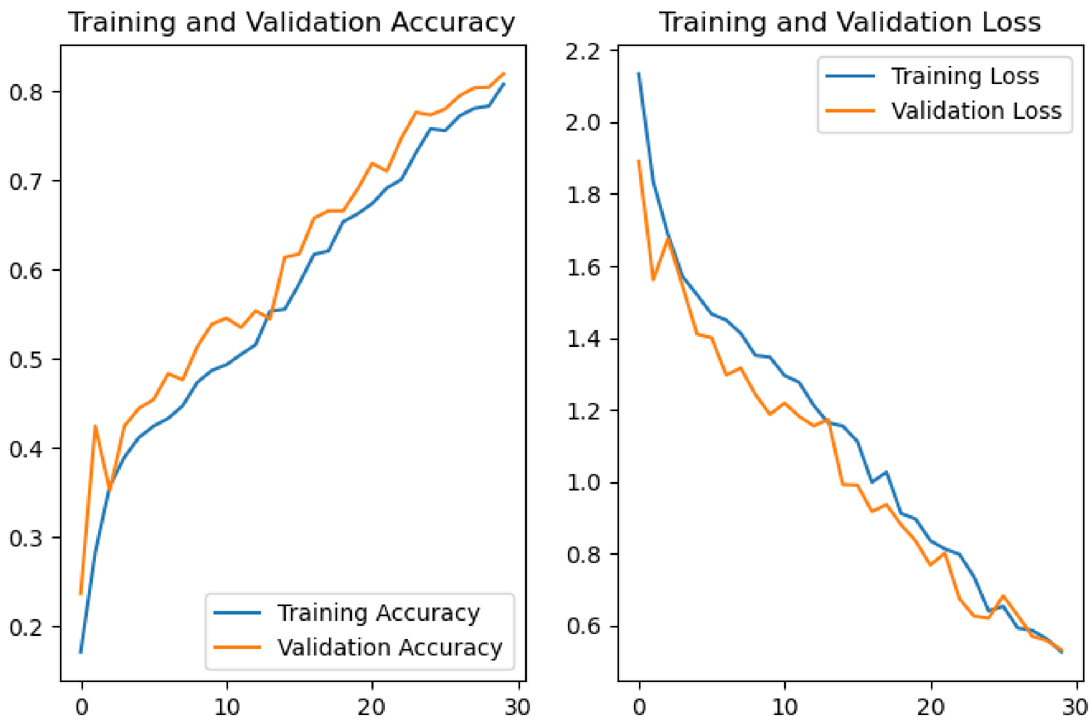
In [151]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

epochs_range = range(epochs)

plt.figure(figsize=(8, 5))
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()
```



Observations

Above charts clearly denotes that model has been able to achieve a good performance on adding augmentation techniques to the existing set of images. However, the model seems to perform well around 20 epoch and post that point , validation accuracy has stopped improving and also the difference between training and validation accuracy is also getting increased after 20 epochs

In [ ]:
```
!pip install tabulate
```

In [152]:
```
from tabulate import tabulate
data = []
for epoch, train_acc, val_acc, train_loss, val_loss in zip(epochs_range, acc,
```

In [152]:
```
from tabulate import tabulate
data = []
for epoch, train_acc, val_acc, train_loss, val_loss in zip(epochs_range, acc,
    data.append([epoch, train_acc, val_acc, train_loss, val_loss])

headers = ["Epochs", "Training Accuracy", "Validation Accuracy", "Training Los

table = tabulate(data, headers, tablefmt="pipe")
print(table)
```

```
|   Epochs |   Training Accuracy |   Validation Accuracy |   Training Loss |
Validation Loss |
|---------:|--------------------:|----------------------:|----------------:|
```

Epochs	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
0	0.171365	0.236823	2.13325	1.89003
1	0.283012	0.424647	1.83361	1.56115
2	0.357567	0.352635	1.68762	1.67549
3	0.389837	0.424647	1.57061	1.54443
4	0.411907	0.444692	1.51949	1.40984
5	0.424518	0.454343	1.46558	1.39974
6	0.433234	0.483296	1.44905	1.29637
7	0.447515	0.476615	1.41215	1.31551
8	0.473665	0.513734	1.3511	1.24251
9	0.487018	0.538975	1.34609	1.18688
10	0.493509	0.545657	1.29508	1.21846
11	0.505007	0.535264	1.27618	1.1816
12	0.51595	0.553823	1.212	1.15483
13	0.553598	0.544915	1.16349	1.17272
14	0.555638	0.613957	1.15454	0.991812
15	0.58457	0.617669	1.11185	0.990138
16	0.617211	0.657758	0.998297	0.917413
17	0.621105	0.665924	1.02655	0.936057
18	0.653746	0.665924	0.911556	0.880281
19	0.662648	0.690423	0.896012	0.835145
20	0.674147	0.719376	0.836209	0.767914
21	0.69158	0.710468	0.812942	0.801254
22	0.701224	0.747587	0.798072	0.674691
23	0.731083	0.77654	0.735296	0.626275
24	0.75816	0.773571	0.640806	0.620557
25	0.755935	0.780252	0.653601	0.68248
26	0.772626	0.7951	0.593155	0.628157
27	0.780972	0.804009	0.586527	0.570094
28	0.783568	0.804751	0.562838	0.55776
29	0.807863	0.819599	0.526773	0.532672

In [153]:

```
def getMaxRecordForParameters(colIndex, tag):
    # Find the maximum value based on colIndex
    max_data = max(data, key=lambda x: x[colIndex])
```

```
0.55776  |
|  29 | 0.807863 |   0.819599 |       0.526773 |
0.532672 |
```

In [153]:
```
def getMaxRecordForParameters(colIndex, tag):
    # Find the maximum value based on colIndex
    max_data = max(data, key=lambda x: x[colIndex])
    max_value = max_data[colIndex]
    epoch_with_max_value = max_data[0]
    print(f"{tag}: {max_value} (Epoch {epoch_with_max_value})")
```

In [154]:
```
# Find the maximum training accuracy and corresponding epoch number
getMaxRecordForParameters(1,"Training Accuracy")
getMaxRecordForParameters(2,"Validation Accuracy")
```

```
Training Accuracy: 0.8078634738922119 (Epoch 29)
Validation Accuracy: 0.8195990920066833 (Epoch 29)
```

```
Training Accuracy: It reaches a  value of approximately 81% after 29 epochs.

Validation Accuracy:  the validation accuracy, which is a measure of the
model's performance on unseen data, reaches around 82%  after epoch 29.

We have a got a very decent model after adding images through augmentation
library.
```

Testing the Model

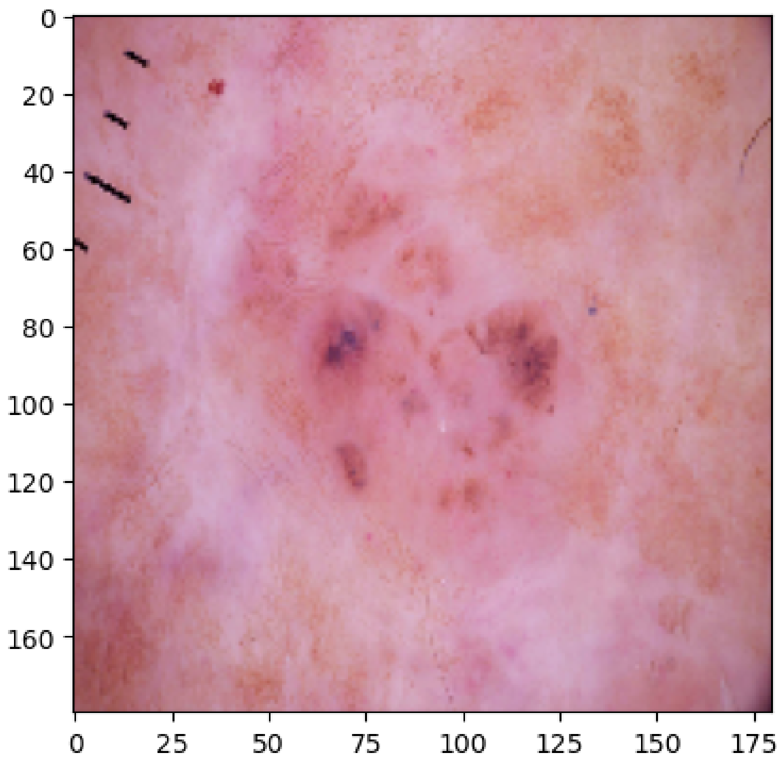
Testing the Model

In [157]:
```
## Though this was not part of the assignment

from glob import glob
Test_image_path = os.path.join(test_images, class_names[1], '*')
Test_image = glob(Test_image_path)
Test_image = load_img(Test_image[-1],target_size=(180,180,3))
plt.imshow(Test_image)
plt.grid(False)

img = np.expand_dims(Test_image,axis=0)
pred = model.predict(img)
pred = np.argmax(pred)
pred_class = class_names[pred]
print("Actual Class "+ class_names[1] +'\n'+ "Predictive Class "+pred_class )
```

```
1/1 [==============================] - 0s 31ms/step
Actual Class basal cell carcinoma
Predictive Class basal cell carcinoma
```



Conclusion

Here is the concluded summary for all the 3 models :

Model 1: Without data augmentation, the model struggled to generalize and had a significant gap between training and validation accuracy.It achieved a training accuracy of 86% but only a validation accuracy of 53%, indicating overfitting.

Model 2:The addition of data augmentation helped reduce overfitting, as seen by the smaller gap between training and validation accuracy.Training accuracy improved to 56%, but overall accuracy only increased marginally.
Model 3:Addressing class imbalance by generating additional data for sparse classes

Model 3:Addressing class imbalance by generating additional data for sparse classes