ozrih35nc

February 17, 2025

0.1 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 that can evaluate images and alert dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

0.1.1 Import the necessary library

[1]:
```
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

[2]:
```
# Define the file paths for the training and testing images
data_dir_train = pathlib.Path(r"C:\Users\fg722f\Documents\Upgrad -␣
↪AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging␣
↪Collaboration\Train")
data_dir_test = pathlib.Path(r"C:\Users\fg722f\Documents\Upgrad -␣
↪AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging␣
↪Collaboration\Test")

print("Train directory exists:", data_dir_train.exists())
print("Test directory exists:", data_dir_test.exists())
```

```
Train directory exists: True
Test directory exists: True
```

[3]:
```
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
```

0.2 Keras - Preprocessing

0.3 Parameter for loader

[4]:
```
batch_size = 32
img_height = 180
img_width = 180
```

0.3.1 Training - dataset

[5]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    seed=123,
    validation_split = 0.2,
    subset='training',
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

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

0.3.2 Validating – dataset

[6]:
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir_train,
    seed=123,
    validation_split = 0.2,
    subset='validation',
    image_size=(img_height, img_width),
    batch_size=batch_size)
```

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

[7]:
```
# Correspond to the directory names in alphabetical order.
class_names = train_ds.class_names
print(class_names)
```

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

0.4 Visualization

[8]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

Dataset.cache() stores images in memory after they are loaded from disk during the first epoch, reducing disk I/O and speeding up training.

Dataset.prefetch() optimizes performance by overlapping data preprocessing with model execution, ensuring efficient data pipeline utilization during training.

[9]:
```
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)
```

0.5 Creating the model

[10]:
```
target_labels = 9

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

```
C:\Users\fg722f\AppData\Roaming\Python\Python311\site-
packages\keras\src\layers\preprocessing\tf_data_layer.py:19: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(**kwargs)
```

[11]:
```
### Model Summary
```

[12]:
```
model.summary()
```

```
Model: "sequential"


 Layer (type)                         Output Shape                          ␣
↪Param #

 rescaling (Rescaling)                (None, 180, 180, 3)                      ␣
↪  0

 conv2d (Conv2D)                      (None, 180, 180, 16)                     ␣
↪448
```

```
max_pooling2d (MaxPooling2D)          (None, 90, 90, 16)                    ␣
↪   0

conv2d_1 (Conv2D)                     (None, 90, 90, 32)                  ␣
↪4,640

max_pooling2d_1 (MaxPooling2D)        (None, 45, 45, 32)                    ␣
↪   0

conv2d_2 (Conv2D)                     (None, 45, 45, 64)                 ␣
↪18,496

max_pooling2d_2 (MaxPooling2D)        (None, 22, 22, 64)                    ␣
↪   0

flatten (Flatten)                     (None, 30976)                         ␣
↪   0

dense (Dense)                         (None, 128)                     ␣
↪3,965,056

dense_1 (Dense)                       (None, 9)                           ␣
↪1,161
```

Total params: 3,989,801 (15.22 MB)

Trainable params: 3,989,801 (15.22 MB)

Non-trainable params: 0 (0.00 B)

0.5.1 Train the model

[13]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

[14]:
```
%%time
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
```

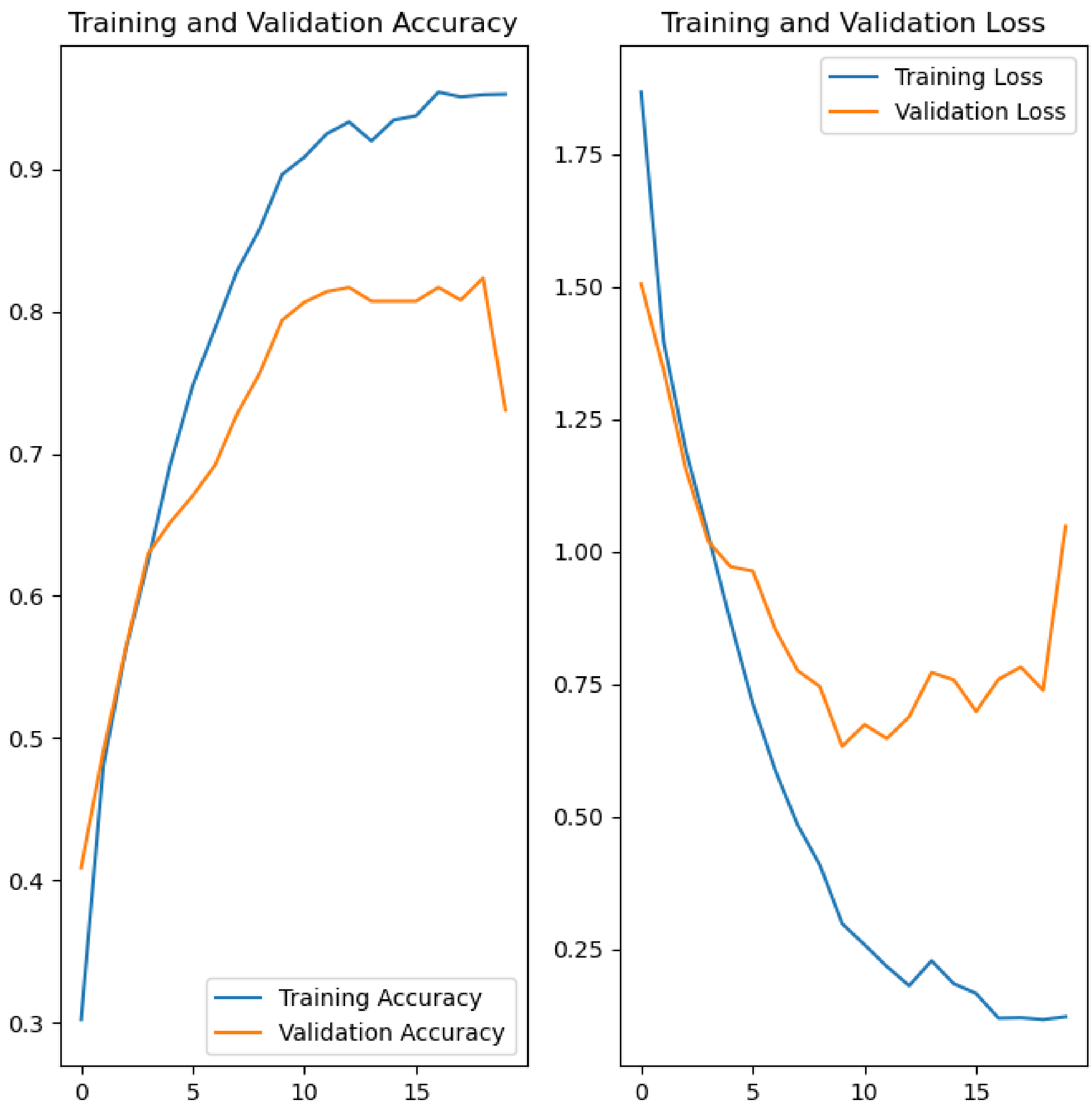
```
)
```

```
Epoch 1/20
169/169          75s 343ms/step -
accuracy: 0.2204 - loss: 2.0690 - val_accuracy: 0.4091 - val_loss: 1.5050
Epoch 2/20
169/169          47s 275ms/step -
accuracy: 0.4657 - loss: 1.4367 - val_accuracy: 0.4915 - val_loss: 1.3438
Epoch 3/20
169/169          48s 286ms/step -
accuracy: 0.5513 - loss: 1.2160 - val_accuracy: 0.5635 - val_loss: 1.1554
Epoch 4/20
169/169          48s 282ms/step -
accuracy: 0.5997 - loss: 1.0728 - val_accuracy: 0.6295 - val_loss: 1.0198
Epoch 5/20
169/169          53s 315ms/step -
accuracy: 0.6964 - loss: 0.8530 - val_accuracy: 0.6518 - val_loss: 0.9718
Epoch 6/20
169/169          49s 288ms/step -
accuracy: 0.7442 - loss: 0.7127 - val_accuracy: 0.6704 - val_loss: 0.9635
Epoch 7/20
169/169          45s 264ms/step -
accuracy: 0.7839 - loss: 0.6018 - val_accuracy: 0.6919 - val_loss: 0.8539
Epoch 8/20
169/169          45s 268ms/step -
accuracy: 0.8268 - loss: 0.4936 - val_accuracy: 0.7283 - val_loss: 0.7759
Epoch 9/20
169/169          47s 276ms/step -
accuracy: 0.8467 - loss: 0.4211 - val_accuracy: 0.7565 - val_loss: 0.7457
Epoch 10/20
169/169          44s 259ms/step -
accuracy: 0.9025 - loss: 0.2865 - val_accuracy: 0.7936 - val_loss: 0.6328
Epoch 11/20
169/169          44s 262ms/step -
accuracy: 0.9136 - loss: 0.2544 - val_accuracy: 0.8062 - val_loss: 0.6738
Epoch 12/20
169/169          51s 304ms/step -
accuracy: 0.9284 - loss: 0.2073 - val_accuracy: 0.8137 - val_loss: 0.6475
Epoch 13/20
169/169          45s 266ms/step -
accuracy: 0.9405 - loss: 0.1646 - val_accuracy: 0.8166 - val_loss: 0.6886
Epoch 14/20
169/169          45s 266ms/step -
accuracy: 0.9299 - loss: 0.1943 - val_accuracy: 0.8070 - val_loss: 0.7723
Epoch 15/20
169/169          49s 288ms/step -
accuracy: 0.9406 - loss: 0.1676 - val_accuracy: 0.8070 - val_loss: 0.7581
```

```
Epoch 16/20
169/169              49s 290ms/step -
accuracy: 0.9466 - loss: 0.1413 - val_accuracy: 0.8070 - val_loss: 0.6985
Epoch 17/20
169/169              47s 275ms/step -
accuracy: 0.9543 - loss: 0.1148 - val_accuracy: 0.8166 - val_loss: 0.7594
Epoch 18/20
169/169              44s 262ms/step -
accuracy: 0.9539 - loss: 0.1168 - val_accuracy: 0.8077 - val_loss: 0.7824
Epoch 19/20
169/169              44s 263ms/step -
accuracy: 0.9558 - loss: 0.1054 - val_accuracy: 0.8233 - val_loss: 0.7386
Epoch 20/20
169/169              45s 265ms/step -
accuracy: 0.9500 - loss: 0.1235 - val_accuracy: 0.7305 - val_loss: 1.0480
CPU times: total: 1h 45min 57s
Wall time: 16min 2s
```

0.5.2 Visualizing the Training Results

[15]:
```
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()
```



0.5.3 Observations:

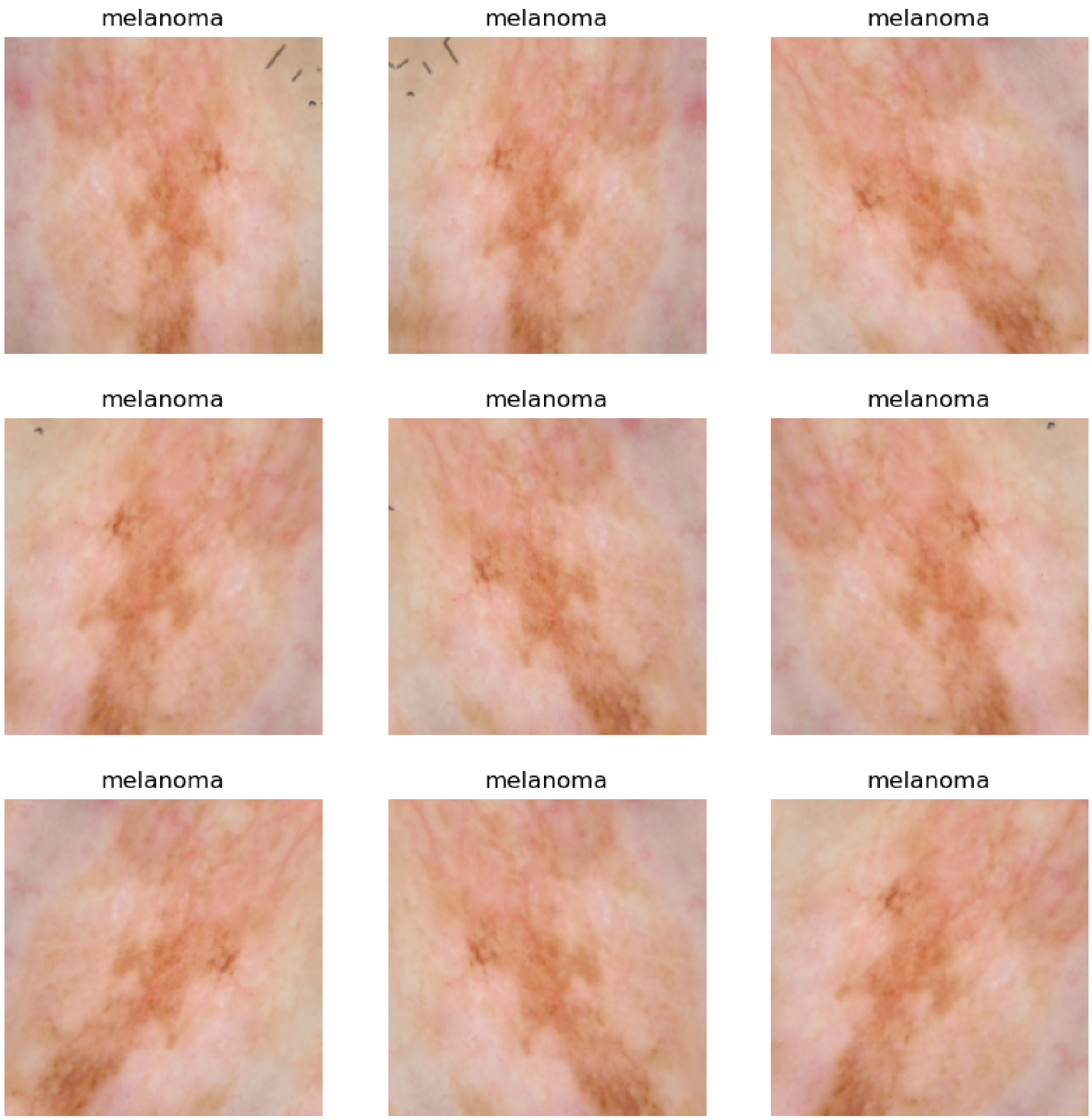
- The model's training accuracy steadily increases to 90%, while validation accuracy remains consistently around 55%.
- The high training accuracy suggests that the model has learned patterns from the training data effectively. However, its poor performance on validation data indicates a lack of generalization, meaning the model is overfitting to the training set.
- To mitigate overfitting, data augmentation techniques will be applied. Given the limited training data, new samples will be generated by introducing slight modifications to existing images, such as horizontal and vertical flips, minor rotations, and other transformations. These augmented images will enhance model robustness and improve its ability to generalize to unseen data.

0.6 After analyzing the model's fit history for signs of underfitting or overfitting, choose an appropriate data augmentation strategy.

[16]:
```
augmentation_data = keras.Sequential([
    layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
])
```

0.6.1 Visualize how your data augmentation strategy applies to a single instance of a training image

[17]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        augmented_images = augmentation_data(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.title(class_names[labels[0]])
        plt.axis("off")
```



0.6.2 Using the Drop Out layer

[18]:
```
model = Sequential([
  augmentation_data,
  layers.Rescaling(1./255),
  layers.Conv2D(16, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
```

```
  layers.Flatten(),
  layers.Dense(128, activation=tf.nn.relu),
  layers.Dense(target_labels)
])
```

0.6.3 Train the model

[19]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

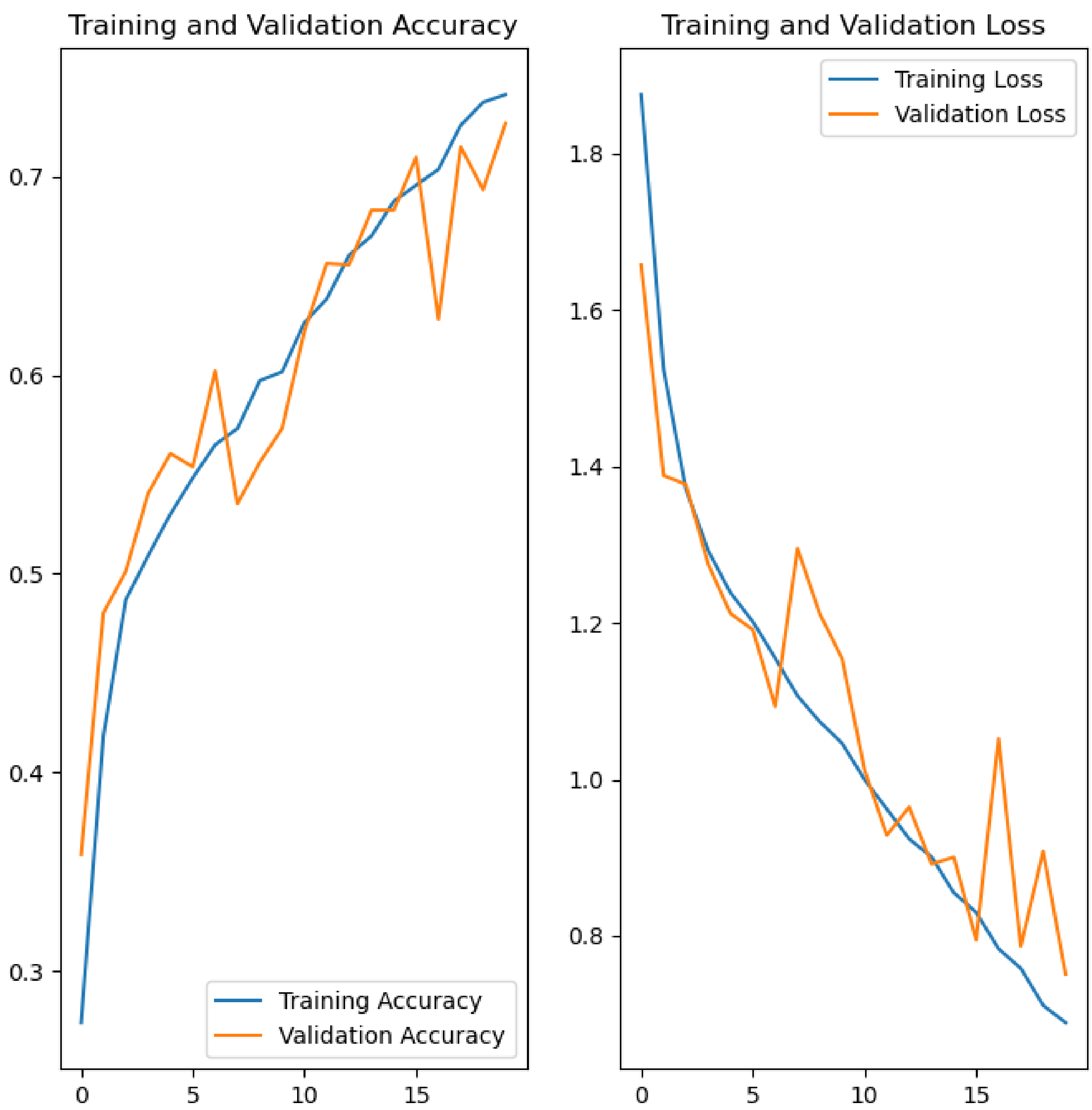
[20]:
```
%%time
## Your code goes here, note: train your model for 20 epochs
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
169/169              54s 300ms/step -
accuracy: 0.2173 - loss: 2.0713 - val_accuracy: 0.3586 - val_loss: 1.6578
Epoch 2/20
169/169              58s 342ms/step -
accuracy: 0.3911 - loss: 1.5863 - val_accuracy: 0.4803 - val_loss: 1.3885
Epoch 3/20
169/169              52s 308ms/step -
accuracy: 0.4731 - loss: 1.3709 - val_accuracy: 0.5011 - val_loss: 1.3772
Epoch 4/20
169/169              51s 304ms/step -
accuracy: 0.5028 - loss: 1.3020 - val_accuracy: 0.5405 - val_loss: 1.2751
Epoch 5/20
169/169              51s 301ms/step -
accuracy: 0.5267 - loss: 1.2543 - val_accuracy: 0.5605 - val_loss: 1.2119
Epoch 6/20
169/169              49s 291ms/step -
accuracy: 0.5511 - loss: 1.1901 - val_accuracy: 0.5538 - val_loss: 1.1914
Epoch 7/20
169/169              50s 295ms/step -
accuracy: 0.5664 - loss: 1.1695 - val_accuracy: 0.6021 - val_loss: 1.0930
Epoch 8/20
169/169              51s 303ms/step -
accuracy: 0.5849 - loss: 1.0810 - val_accuracy: 0.5353 - val_loss: 1.2951
Epoch 9/20
169/169              49s 292ms/step -
accuracy: 0.5853 - loss: 1.0827 - val_accuracy: 0.5561 - val_loss: 1.2114
```

```
Epoch 10/20
169/169            50s 294ms/step -
accuracy: 0.5967 - loss: 1.0344 - val_accuracy: 0.5731 - val_loss: 1.1541
Epoch 11/20
169/169            49s 292ms/step -
accuracy: 0.6100 - loss: 1.0188 - val_accuracy: 0.6221 - val_loss: 1.0121
Epoch 12/20
169/169            51s 299ms/step -
accuracy: 0.6546 - loss: 0.9204 - val_accuracy: 0.6563 - val_loss: 0.9286
Epoch 13/20
169/169            53s 312ms/step -
accuracy: 0.6671 - loss: 0.9106 - val_accuracy: 0.6555 - val_loss: 0.9647
Epoch 14/20
169/169            53s 315ms/step -
accuracy: 0.6766 - loss: 0.8835 - val_accuracy: 0.6830 - val_loss: 0.8916
Epoch 15/20
169/169            50s 298ms/step -
accuracy: 0.6942 - loss: 0.8433 - val_accuracy: 0.6830 - val_loss: 0.9005
Epoch 16/20
169/169            51s 302ms/step -
accuracy: 0.6939 - loss: 0.8145 - val_accuracy: 0.7097 - val_loss: 0.7947
Epoch 17/20
169/169            55s 325ms/step -
accuracy: 0.7044 - loss: 0.7670 - val_accuracy: 0.6281 - val_loss: 1.0519
Epoch 18/20
169/169            59s 351ms/step -
accuracy: 0.7203 - loss: 0.7562 - val_accuracy: 0.7149 - val_loss: 0.7869
Epoch 19/20
169/169            52s 308ms/step -
accuracy: 0.7382 - loss: 0.7264 - val_accuracy: 0.6934 - val_loss: 0.9080
Epoch 20/20
169/169            54s 319ms/step -
accuracy: 0.7350 - loss: 0.6952 - val_accuracy: 0.7268 - val_loss: 0.7510
CPU times: total: 1h 56min 34s
Wall time: 17min 22s
```

[21]:
```python
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()
```



0.6.4 Observations:

- By utilizing augmented data, the issue of overfitting has been effectively mitigated.
- The training and validation accuracy now fall within a similar range, suggesting improved generalization.
- However, both the training and validation accuracies are showing poor performance, indicating that the model is now underfitting.

[22]:
```
from glob import glob

## find the image path for all class labels (lesions)
images_path_list = [ i for i in glob(os.path.join(data_dir_train, '*', '*.
↪jpg')) ]

lesions_list = [ os.path.basename(os.path.dirname(j)) for j in glob(os.path.
↪join(data_dir_train, '*', '*.jpg')) ]
print(len(lesions_list))
```

```
2239
```

[23]:
```
# Extract image path and class label in a dictionary
image_dict = dict(zip(images_path_list, lesions_list))
print(list(image_dict.items())[:5])
```

```
[('C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin
cancer ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\ISIC_0025780.jpg', 'actinic keratosis'),
('C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\ISIC_0025803.jpg', 'actinic keratosis'),
('C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\ISIC_0025825.jpg', 'actinic keratosis'),
('C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\ISIC_0025953.jpg', 'actinic keratosis'),
('C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\ISIC_0025957.jpg', 'actinic keratosis')]
```

[24]:
```
# View the image paths and corresponding class labels in a DataFrame
lesions_df = pd.DataFrame(list(image_dict.items()), columns=['Image Path',␣
↪'Label'])
lesions_df.head()
```

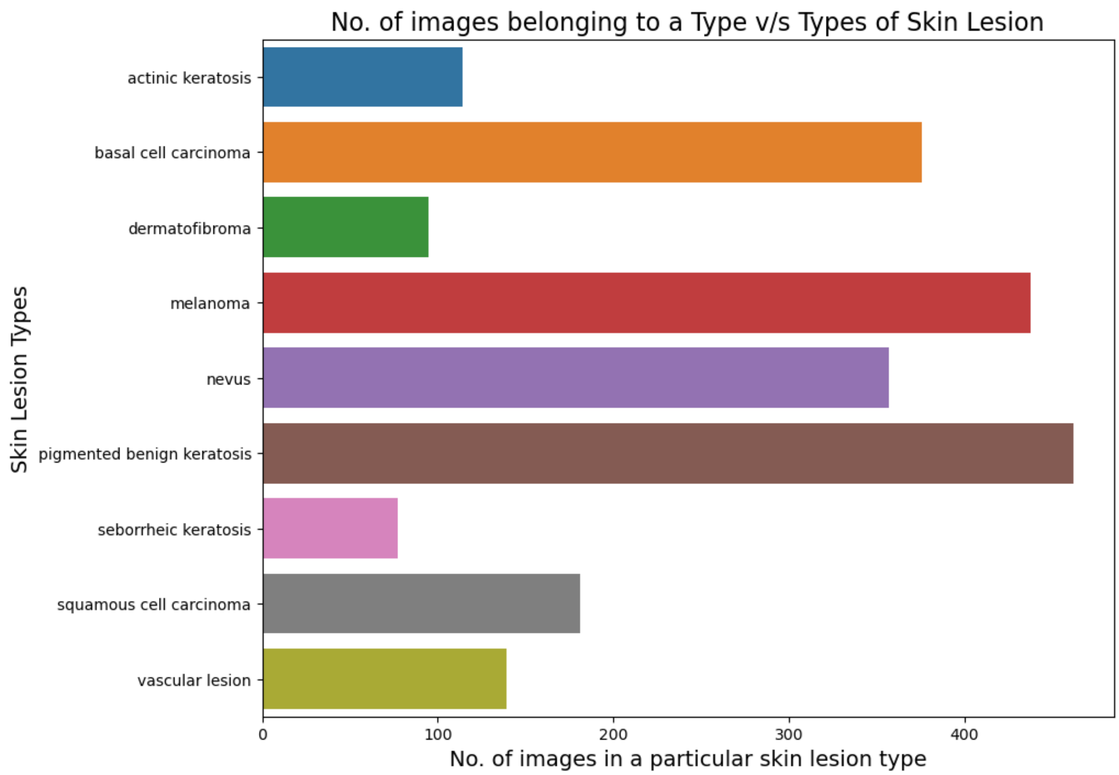
[24]:
```
                                          Image Path              Label
0  C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…  actinic keratosis
```

```
1  C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…  actinic keratosis
2  C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…  actinic keratosis
3  C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…  actinic keratosis
4  C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…  actinic keratosis
```

[25]:
```
## Inspecting the class distribution in the dataset
lesions_df[['Label']].value_counts()
```

[25]:
```
Label
pigmented benign keratosis    462
melanoma                      438
basal cell carcinoma          376
nevus                         357
squamous cell carcinoma       181
vascular lesion               139
actinic keratosis             114
dermatofibroma                 95
seborrheic keratosis           77
Name: count, dtype: int64
```

[26]:
```
# Visualize the distribution of classes using a countplot

import seaborn as sns

plt.figure(figsize=(10, 8))
sns.countplot(y="Label", data=lesions_df)
plt.title('No. of images belonging to a Type v/s Types of Skin Lesion',␣
 ↪fontsize=16)
plt.xlabel('No. of images in a particular skin lesion type', fontsize=14)
plt.ylabel('Skin Lesion Types', fontsize=14)
plt.show()
```



[27]:
```
round(lesions_df[['Label']].value_counts(normalize=True)*100, 2)
```

[27]:
```
Label
pigmented benign keratosis    20.63
melanoma                      19.56
basal cell carcinoma          16.79
nevus                         15.94
squamous cell carcinoma        8.08
vascular lesion                6.21
actinic keratosis              5.09
dermatofibroma                 4.24
seborrheic keratosis           3.44
Name: proportion, dtype: float64
```

0.6.5 Observations:

- A clear class imbalance is observed in the training data.
- The class **"seborrheic keratosis"** constitutes the smallest proportion of samples, making up approximately **3.44%**.
- In contrast, the classes **"pigmented benign keratosis"** and **"melanoma"** dominate the dataset, representing approximately **20.63%** and **19.56%** of the data, respectively.

[28]:

```
path_to_training_dataset = str(data_dir_train) + '/'

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)
```

```
Initialised with 114 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/actinic keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x27891883590>:
100%| | 500/500 [00:06<00:00, 83.19 Samples/

Initialised with 376 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/basal cell carcinoma\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x278D62B8410>:
100%| | 500/500 [00:06<00:00, 75.73 Samples/

Initialised with 95 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/dermatofibroma\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x278D6276390>:
100%| | 500/500 [00:07<00:00, 71.41 Samples/

Initialised with 438 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/melanoma\output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1024x768 at
0x278D5DE2150>: 100%| | 500/500 [00:28<00

Initialised with 357 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/nevus\output.

Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x278D34616D0>:
100%| | 500/500 [00:25<00:00, 20.00 Samples

Initialised with 462 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
```

```
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/pigmented benign keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x27891865C10>:
100%| | 500/500 [00:06<00:00, 73.01 Samples/

Initialised with 77 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/seborrheic keratosis\output.

Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x278D139CD90>:
100%| | 500/500 [00:13<00:00, 36.27 Samples

Initialised with 181 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/squamous cell carcinoma\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x278D353BC50>:
100%| | 500/500 [00:06<00:00, 76.83 Samples/

Initialised with 139 image(s) found.
Output directory set to C:\Users\fg722f\Documents\Upgrad -
AI,ML\CNN\CNN_assignment\Skin cancer ISIC The International Skin Imaging
Collaboration\Train/vascular lesion\output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x278D5E148D0>:
100%| | 500/500 [00:06<00:00, 78.91 Samples/
```

[29]:
```
# Verifying the total count of images after the augmentation
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg')))
print(image_count_train)
```

```
9000
```

0.6.6 Let's examine the distribution of the augmented data after adding new images to the original training dataset

[30]:
```
# extracting the augmented image paths in a list
path_list_new = [x for x in glob(os.path.join(data_dir_train, '*','output', '*.
↪jpg'))]
path_list_new[:5]
```

[30]:
```
['C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\output\\actinic
keratosis_original_ISIC_0025780.jpg_19df1b64-294e-4f75-9187-e6390dcc0b2f.jpg',
 'C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\output\\actinic
```

```
keratosis_original_ISIC_0025780.jpg_60ffb8bb-a54e-4b34-8673-99c796b68c43.jpg',
 'C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\output\\actinic
keratosis_original_ISIC_0025780.jpg_77f997c2-8f25-4d27-a3d7-7047cf4317df.jpg',
 'C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\output\\actinic
keratosis_original_ISIC_0025780.jpg_9d697ca1-517f-407f-a12a-7da83248a9d9.jpg',
 'C:\\Users\\fg722f\\Documents\\Upgrad - AI,ML\\CNN\\CNN_assignment\\Skin cancer
ISIC The International Skin Imaging Collaboration\\Train\\actinic
keratosis\\output\\actinic
keratosis_original_ISIC_0025780.jpg_c17716cd-5770-42be-9d10-2d829951ed0c.jpg']
```

[31]:
```
lesion_list_new = [os.path.basename(os.path.dirname(os.path.dirname(y))) for y␣
 ↪in glob(os.path.join(data_dir_train, '*','output', '*.jpg'))]
lesion_list_new[:5]
```

[31]:
```
['actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis',
 'actinic keratosis']
```

[32]:
```
dataframe_dict_new = dict(zip(path_list_new, lesion_list_new))
```

[33]:
```
df2 = pd.DataFrame(list(dataframe_dict_new.items()),columns = ['Image␣
 ↪Path','Label'])
new_df = pd.concat([lesions_df, df2], ignore_index=True)
new_df.shape
```

[33]:
```
(11239, 2)
```

[34]:
```
new_df.head()
```

[34]:

	Image Path	Label
0	C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…	actinic keratosis
1	C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…	actinic keratosis
2	C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…	actinic keratosis
3	C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…	actinic keratosis
4	C:\Users\fg722f\Documents\Upgrad - AI,ML\CNN\C…	actinic keratosis

[35]:
```
# Inspecting the classes after adding 500 samples per label
new_df['Label'].value_counts()
```

[35]:
```
Label
pigmented benign keratosis    1462
```

```
melanoma                      1438
basal cell carcinoma          1376
nevus                         1357
squamous cell carcinoma       1181
vascular lesion               1139
actinic keratosis             1114
dermatofibroma                1095
seborrheic keratosis          1077
Name: count, dtype: int64
```

[36]:
```python
# Inspecting the classes (% age wise) after adding 500 samples per label
round(new_df['Label'].value_counts(normalize=True)*100, 2)
```

[36]:
```
Label
pigmented benign keratosis    13.01
melanoma                      12.79
basal cell carcinoma          12.24
nevus                         12.07
squamous cell carcinoma       10.51
vascular lesion               10.13
actinic keratosis              9.91
dermatofibroma                 9.74
seborrheic keratosis           9.58
Name: proportion, dtype: float64
```

0.6.7 Train the model on the data created using Augmentor

[37]:
```python
batch_size = 32
img_height = 180
img_width = 180
```

0.6.8 Training dataset

[38]:
```python
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  validation_split = 0.2,
  subset = 'training',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

0.6.9 Validation dataset

[39]:
```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  validation_split = 0.2,
  subset = 'validation',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

0.6.10 Model with Normalization

[40]:
```
model = Sequential([
  augmentation_data,
  layers.Rescaling(1./255),
  layers.Conv2D(16, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation=tf.nn.relu),
  layers.Dense(target_labels)
])
```

0.6.11 Train the model

[41]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

[42]:
```
%%time
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
281/281              210s 719ms/step -
accuracy: 0.2732 - loss: 3.0251 - val_accuracy: 0.1718 - val_loss: 16.9215
Epoch 2/20
281/281              199s 709ms/step -
accuracy: 0.4198 - loss: 1.5003 - val_accuracy: 0.4299 - val_loss: 1.6181
Epoch 3/20
281/281              180s 639ms/step -
accuracy: 0.4680 - loss: 1.3945 - val_accuracy: 0.4878 - val_loss: 1.3198
Epoch 4/20
281/281              191s 678ms/step -
accuracy: 0.4992 - loss: 1.2904 - val_accuracy: 0.5162 - val_loss: 1.1567
Epoch 5/20
281/281              184s 652ms/step -
accuracy: 0.5216 - loss: 1.2027 - val_accuracy: 0.5256 - val_loss: 1.1751
Epoch 6/20
281/281              170s 604ms/step -
accuracy: 0.5537 - loss: 1.1439 - val_accuracy: 0.5652 - val_loss: 1.1432
Epoch 7/20
281/281              173s 614ms/step -
accuracy: 0.5739 - loss: 1.0940 - val_accuracy: 0.5897 - val_loss: 1.0724
Epoch 8/20
281/281              179s 637ms/step -
accuracy: 0.5986 - loss: 1.0459 - val_accuracy: 0.5670 - val_loss: 1.1598
Epoch 9/20
281/281              192s 682ms/step -
accuracy: 0.6148 - loss: 1.0053 - val_accuracy: 0.5781 - val_loss: 1.1375
Epoch 10/20
281/281              178s 633ms/step -
accuracy: 0.6405 - loss: 0.9498 - val_accuracy: 0.5470 - val_loss: 1.4007
Epoch 11/20
281/281              168s 595ms/step -
accuracy: 0.6477 - loss: 0.9279 - val_accuracy: 0.5670 - val_loss: 1.3061
Epoch 12/20
281/281              174s 619ms/step -
accuracy: 0.6592 - loss: 0.8923 - val_accuracy: 0.6542 - val_loss: 0.9055
Epoch 13/20
281/281              175s 623ms/step -
accuracy: 0.6749 - loss: 0.8551 - val_accuracy: 0.6253 - val_loss: 0.9753
Epoch 14/20
281/281              179s 636ms/step -
accuracy: 0.6886 - loss: 0.8270 - val_accuracy: 0.6551 - val_loss: 0.9287
Epoch 15/20
281/281              204s 726ms/step -
accuracy: 0.6947 - loss: 0.8013 - val_accuracy: 0.4246 - val_loss: 2.7548
Epoch 16/20
281/281              247s 877ms/step -
accuracy: 0.6964 - loss: 0.8012 - val_accuracy: 0.5817 - val_loss: 1.2504
```

```
Epoch 17/20
281/281              247s 878ms/step -
accuracy: 0.7204 - loss: 0.7440 - val_accuracy: 0.6640 - val_loss: 0.9198
Epoch 18/20
281/281              246s 876ms/step -
accuracy: 0.7310 - loss: 0.7281 - val_accuracy: 0.6627 - val_loss: 0.9579
Epoch 19/20
281/281              246s 875ms/step -
accuracy: 0.7233 - loss: 0.7161 - val_accuracy: 0.7486 - val_loss: 0.6845
Epoch 20/20
281/281              245s 872ms/step -
accuracy: 0.7360 - loss: 0.6961 - val_accuracy: 0.4495 - val_loss: 2.6095
CPU times: total: 7h 39min 10s
Wall time: 1h 6min 27s
```

[43]:

```
model = Sequential([
  augmentation_data,
  layers.Rescaling(1./255),
  layers.Conv2D(16, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3, 3), padding='same', activation=tf.nn.relu),
  layers.BatchNormalization(),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation=tf.nn.relu),
  layers.Dense(target_labels)
])
```

0.6.12 Train the Model

[44]:

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

[45]:

```
%%time
epochs = 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
281/281          260s 875ms/step -
accuracy: 0.2878 - loss: 3.5128 - val_accuracy: 0.1509 - val_loss: 22.5871
Epoch 2/20
281/281          245s 873ms/step -
accuracy: 0.3617 - loss: 1.6663 - val_accuracy: 0.3111 - val_loss: 2.4115
Epoch 3/20
281/281          241s 857ms/step -
accuracy: 0.4158 - loss: 1.5032 - val_accuracy: 0.4597 - val_loss: 1.3610
Epoch 4/20
281/281          241s 858ms/step -
accuracy: 0.4675 - loss: 1.3447 - val_accuracy: 0.4032 - val_loss: 1.5602
Epoch 5/20
281/281          247s 879ms/step -
accuracy: 0.4896 - loss: 1.2744 - val_accuracy: 0.4793 - val_loss: 1.5388
Epoch 6/20
281/281          249s 886ms/step -
accuracy: 0.5215 - loss: 1.2155 - val_accuracy: 0.5189 - val_loss: 1.2496
Epoch 7/20
281/281          240s 852ms/step -
accuracy: 0.5687 - loss: 1.1274 - val_accuracy: 0.1985 - val_loss: 3.0677
Epoch 8/20
281/281          252s 895ms/step -
accuracy: 0.5666 - loss: 1.1021 - val_accuracy: 0.4709 - val_loss: 1.4528
Epoch 9/20
281/281          250s 889ms/step -
accuracy: 0.5872 - loss: 1.0629 - val_accuracy: 0.6012 - val_loss: 0.9923
Epoch 10/20
281/281          240s 855ms/step -
accuracy: 0.6147 - loss: 0.9983 - val_accuracy: 0.5594 - val_loss: 1.2790
Epoch 11/20
281/281          238s 845ms/step -
accuracy: 0.6262 - loss: 0.9588 - val_accuracy: 0.4588 - val_loss: 1.6419
Epoch 12/20
281/281          245s 872ms/step -
accuracy: 0.6432 - loss: 0.9237 - val_accuracy: 0.2737 - val_loss: 4.4214
Epoch 13/20
281/281          247s 879ms/step -
accuracy: 0.6525 - loss: 0.8952 - val_accuracy: 0.4722 - val_loss: 1.5996
Epoch 14/20
281/281          213s 760ms/step -
accuracy: 0.6689 - loss: 0.8717 - val_accuracy: 0.5336 - val_loss: 1.5008
Epoch 15/20
281/281          218s 775ms/step -
accuracy: 0.6891 - loss: 0.8268 - val_accuracy: 0.5852 - val_loss: 1.1112
Epoch 16/20
281/281          184s 654ms/step -
accuracy: 0.6877 - loss: 0.8202 - val_accuracy: 0.6253 - val_loss: 0.9688
```

```
Epoch 17/20
281/281              176s 626ms/step -
accuracy: 0.7049 - loss: 0.7761 - val_accuracy: 0.6547 - val_loss: 0.9237
Epoch 18/20
281/281              175s 621ms/step -
accuracy: 0.7210 - loss: 0.7470 - val_accuracy: 0.5238 - val_loss: 1.8793
Epoch 19/20
281/281              181s 644ms/step -
accuracy: 0.7257 - loss: 0.7372 - val_accuracy: 0.5523 - val_loss: 1.4420
Epoch 20/20
281/281              193s 687ms/step -
accuracy: 0.7328 - loss: 0.7045 - val_accuracy: 0.4722 - val_loss: 2.2976
CPU times: total: 8h 40min 21s
Wall time: 1h 15min 36s
```

0.6.13 Model Summary

[46]:
```
model.summary()
```

```
Model: "sequential_4"


 Layer (type)                       Output Shape                          ␣
↪Param #

 sequential_1 (Sequential)          (None, 180, 180, 3)                     ␣
↪  0

 rescaling_3 (Rescaling)            (None, 180, 180, 3)                     ␣
↪  0

 conv2d_9 (Conv2D)                  (None, 180, 180, 16)                    ␣
↪448

 batch_normalization_3              (None, 180, 180, 16)                    ␣
↪ 64
 (BatchNormalization)                                                       ␣
↪

 max_pooling2d_9 (MaxPooling2D)     (None, 90, 90, 16)                      ␣
↪  0

 conv2d_10 (Conv2D)                 (None, 90, 90, 32)                    ␣
↪4,640

 batch_normalization_4              (None, 90, 90, 32)                      ␣
↪128
```

```
(BatchNormalization)                                                   ␣
↪

max_pooling2d_10 (MaxPooling2D)       (None, 45, 45, 32)               ␣
↪  0

conv2d_11 (Conv2D)                    (None, 45, 45, 64)             ␣
↪18,496

batch_normalization_5                 (None, 45, 45, 64)               ␣
↪256
(BatchNormalization)                                                   ␣
↪

max_pooling2d_11 (MaxPooling2D)       (None, 22, 22, 64)               ␣
↪  0

dropout_2 (Dropout)                   (None, 22, 22, 64)               ␣
↪  0

flatten_3 (Flatten)                   (None, 30976)                    ␣
↪  0

dense_6 (Dense)                       (None, 128)                  ␣
↪3,965,056

dense_7 (Dense)                       (None, 9)                       ␣
↪1,161
```

Total params: 11,970,301 (45.66 MB)

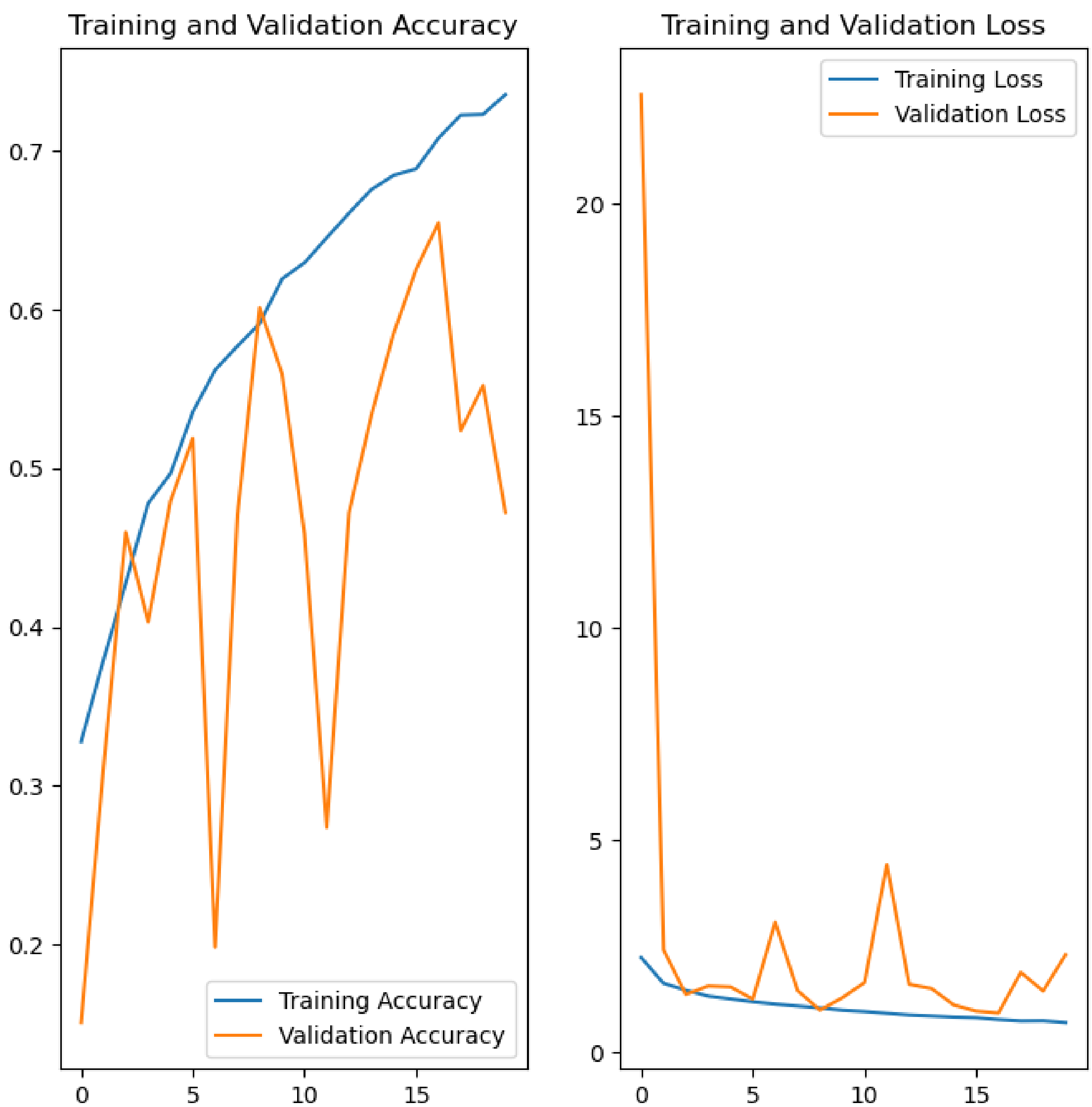
Trainable params: 3,990,025 (15.22 MB)

Non-trainable params: 224 (896.00 B)

Optimizer params: 7,980,052 (30.44 MB)

[47]:
```
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()
```



0.6.14 Create the model without Batch Normalization

[48]:
```
model = Sequential([
  augmentation_data,
  layers.Rescaling(1./255),
  layers.Conv2D(16, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Conv2D(32, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Conv2D(64, (3, 3), padding='same', activation=tf.nn.relu),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation=tf.nn.relu),
  layers.Dense(target_labels)
])
```

[49]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
checkpoint = ModelCheckpoint("model.keras", monitor="val_accuracy",␣
 ↪save_best_only=True, mode="auto", verbose=1)

# Early stop the training when a monitored metric ceases to show improvement
earlystop = EarlyStopping(monitor="val_accuracy", patience=5, mode="auto",␣
 ↪verbose=1)
```

0.6.15 Train the model

[50]:
```
%%time

epochs = 50
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  callbacks=[checkpoint, earlystop]
)
```

```
Epoch 1/50
```

```
281/281              0s 366ms/step -
accuracy: 0.2516 - loss: 1.9658
Epoch 1: val_accuracy improved from -inf to 0.48020, saving model to model.keras
281/281              118s 402ms/step -
accuracy: 0.2519 - loss: 1.9649 - val_accuracy: 0.4802 - val_loss: 1.4744
Epoch 2/50
281/281              0s 467ms/step -
accuracy: 0.4542 - loss: 1.4304
Epoch 2: val_accuracy improved from 0.48020 to 0.53004, saving model to
model.keras
281/281              144s 511ms/step -
accuracy: 0.4542 - loss: 1.4302 - val_accuracy: 0.5300 - val_loss: 1.2622
Epoch 3/50
281/281              0s 422ms/step -
accuracy: 0.5068 - loss: 1.3026
Epoch 3: val_accuracy improved from 0.53004 to 0.56030, saving model to
model.keras
281/281              129s 457ms/step -
accuracy: 0.5068 - loss: 1.3025 - val_accuracy: 0.5603 - val_loss: 1.1738
Epoch 4/50
281/281              0s 436ms/step -
accuracy: 0.5430 - loss: 1.1980
Epoch 4: val_accuracy improved from 0.56030 to 0.56698, saving model to
model.keras
281/281              137s 487ms/step -
accuracy: 0.5431 - loss: 1.1979 - val_accuracy: 0.5670 - val_loss: 1.1031
Epoch 5/50
281/281              0s 616ms/step -
accuracy: 0.5580 - loss: 1.1605
Epoch 5: val_accuracy improved from 0.56698 to 0.62483, saving model to
model.keras
281/281              185s 658ms/step -
accuracy: 0.5580 - loss: 1.1603 - val_accuracy: 0.6248 - val_loss: 0.9862
Epoch 6/50
281/281              0s 356ms/step -
accuracy: 0.6108 - loss: 1.0452
Epoch 6: val_accuracy improved from 0.62483 to 0.65020, saving model to
model.keras
281/281              109s 386ms/step -
accuracy: 0.6108 - loss: 1.0451 - val_accuracy: 0.6502 - val_loss: 0.9425
Epoch 7/50
281/281              0s 433ms/step -
accuracy: 0.6342 - loss: 0.9823
Epoch 7: val_accuracy improved from 0.65020 to 0.67601, saving model to
model.keras
281/281              133s 473ms/step -
accuracy: 0.6342 - loss: 0.9822 - val_accuracy: 0.6760 - val_loss: 0.8536
Epoch 8/50
```

```
281/281            0s 477ms/step -
accuracy: 0.6598 - loss: 0.9158
Epoch 8: val_accuracy improved from 0.67601 to 0.68536, saving model to
model.keras
281/281            146s 517ms/step -
accuracy: 0.6598 - loss: 0.9157 - val_accuracy: 0.6854 - val_loss: 0.8339
Epoch 9/50
281/281            0s 577ms/step -
accuracy: 0.6877 - loss: 0.8491
Epoch 9: val_accuracy did not improve from 0.68536
281/281            177s 630ms/step -
accuracy: 0.6877 - loss: 0.8491 - val_accuracy: 0.6809 - val_loss: 0.8440
Epoch 10/50
281/281            0s 786ms/step -
accuracy: 0.7076 - loss: 0.7951
Epoch 10: val_accuracy improved from 0.68536 to 0.70227, saving model to
model.keras
281/281            239s 851ms/step -
accuracy: 0.7076 - loss: 0.7951 - val_accuracy: 0.7023 - val_loss: 0.8239
Epoch 11/50
281/281            0s 893ms/step -
accuracy: 0.7220 - loss: 0.7428
Epoch 11: val_accuracy improved from 0.70227 to 0.73787, saving model to
model.keras
281/281            279s 990ms/step -
accuracy: 0.7220 - loss: 0.7428 - val_accuracy: 0.7379 - val_loss: 0.7066
Epoch 12/50
281/281            0s 1s/step -
accuracy: 0.7181 - loss: 0.7395
Epoch 12: val_accuracy did not improve from 0.73787
281/281            399s 1s/step -
accuracy: 0.7181 - loss: 0.7394 - val_accuracy: 0.7174 - val_loss: 0.7959
Epoch 13/50
281/281            0s 1s/step -
accuracy: 0.7406 - loss: 0.7123
Epoch 13: val_accuracy did not improve from 0.73787
281/281            371s 1s/step -
accuracy: 0.7407 - loss: 0.7122 - val_accuracy: 0.7303 - val_loss: 0.7832
Epoch 14/50
281/281            0s 1s/step -
accuracy: 0.7652 - loss: 0.6359
Epoch 14: val_accuracy improved from 0.73787 to 0.78772, saving model to
model.keras
281/281            378s 1s/step -
accuracy: 0.7653 - loss: 0.6358 - val_accuracy: 0.7877 - val_loss: 0.6011
Epoch 15/50
281/281            0s 1s/step -
accuracy: 0.7722 - loss: 0.6022
```

```
Epoch 15: val_accuracy did not improve from 0.78772
281/281             354s 1s/step -
accuracy: 0.7722 - loss: 0.6022 - val_accuracy: 0.7824 - val_loss: 0.6164
Epoch 16/50
281/281             0s 1s/step -
accuracy: 0.7786 - loss: 0.5805
Epoch 16: val_accuracy did not improve from 0.78772
281/281             362s 1s/step -
accuracy: 0.7786 - loss: 0.5804 - val_accuracy: 0.7739 - val_loss: 0.6255
Epoch 17/50
281/281             0s 1s/step -
accuracy: 0.7905 - loss: 0.5603
Epoch 17: val_accuracy improved from 0.78772 to 0.79083, saving model to
model.keras
281/281             420s 1s/step -
accuracy: 0.7905 - loss: 0.5603 - val_accuracy: 0.7908 - val_loss: 0.5648
Epoch 18/50
281/281             0s 1s/step -
accuracy: 0.8104 - loss: 0.5203
Epoch 18: val_accuracy did not improve from 0.79083
281/281             439s 2s/step -
accuracy: 0.8104 - loss: 0.5203 - val_accuracy: 0.7833 - val_loss: 0.5951
Epoch 19/50
281/281             0s 1s/step -
accuracy: 0.7903 - loss: 0.5540
Epoch 19: val_accuracy improved from 0.79083 to 0.81353, saving model to
model.keras
281/281             427s 2s/step -
accuracy: 0.7904 - loss: 0.5539 - val_accuracy: 0.8135 - val_loss: 0.5559
Epoch 20/50
281/281             0s 1s/step -
accuracy: 0.8112 - loss: 0.5088
Epoch 20: val_accuracy improved from 0.81353 to 0.81575, saving model to
model.keras
281/281             416s 1s/step -
accuracy: 0.8112 - loss: 0.5088 - val_accuracy: 0.8158 - val_loss: 0.5277
Epoch 21/50
281/281             0s 1s/step -
accuracy: 0.8160 - loss: 0.4767
Epoch 21: val_accuracy improved from 0.81575 to 0.81842, saving model to
model.keras
281/281             433s 2s/step -
accuracy: 0.8160 - loss: 0.4766 - val_accuracy: 0.8184 - val_loss: 0.5158
Epoch 22/50
281/281             0s 785ms/step -
accuracy: 0.8299 - loss: 0.4649
Epoch 22: val_accuracy did not improve from 0.81842
281/281             228s 807ms/step -
```

```
accuracy: 0.8299 - loss: 0.4649 - val_accuracy: 0.8140 - val_loss: 0.5418
Epoch 23/50
281/281            0s 510ms/step -
accuracy: 0.8412 - loss: 0.4353
Epoch 23: val_accuracy did not improve from 0.81842
281/281            155s 552ms/step -
accuracy: 0.8413 - loss: 0.4353 - val_accuracy: 0.8024 - val_loss: 0.5719
Epoch 24/50
281/281            0s 511ms/step -
accuracy: 0.8539 - loss: 0.3958
Epoch 24: val_accuracy did not improve from 0.81842
281/281            156s 553ms/step -
accuracy: 0.8539 - loss: 0.3958 - val_accuracy: 0.8166 - val_loss: 0.5457
Epoch 25/50
281/281            0s 520ms/step -
accuracy: 0.8509 - loss: 0.4075
Epoch 25: val_accuracy improved from 0.81842 to 0.82644, saving model to
model.keras
281/281            159s 566ms/step -
accuracy: 0.8509 - loss: 0.4075 - val_accuracy: 0.8264 - val_loss: 0.5003
Epoch 26/50
281/281            0s 523ms/step -
accuracy: 0.8589 - loss: 0.3849
Epoch 26: val_accuracy improved from 0.82644 to 0.84824, saving model to
model.keras
281/281            157s 557ms/step -
accuracy: 0.8589 - loss: 0.3849 - val_accuracy: 0.8482 - val_loss: 0.4568
Epoch 27/50
281/281            0s 348ms/step -
accuracy: 0.8219 - loss: 0.4844
Epoch 27: val_accuracy improved from 0.84824 to 0.86782, saving model to
model.keras
281/281            106s 379ms/step -
accuracy: 0.8219 - loss: 0.4842 - val_accuracy: 0.8678 - val_loss: 0.4162
Epoch 28/50
281/281            0s 368ms/step -
accuracy: 0.8556 - loss: 0.4083
Epoch 28: val_accuracy did not improve from 0.86782
281/281            112s 396ms/step -
accuracy: 0.8556 - loss: 0.4083 - val_accuracy: 0.8474 - val_loss: 0.4330
Epoch 29/50
281/281            0s 369ms/step -
accuracy: 0.8745 - loss: 0.3517
Epoch 29: val_accuracy did not improve from 0.86782
281/281            112s 399ms/step -
accuracy: 0.8745 - loss: 0.3517 - val_accuracy: 0.8558 - val_loss: 0.4487
Epoch 30/50
281/281            0s 373ms/step -
```

```
accuracy: 0.8779 - loss: 0.3405
Epoch 30: val_accuracy did not improve from 0.86782
281/281             113s 401ms/step -
accuracy: 0.8778 - loss: 0.3406 - val_accuracy: 0.8474 - val_loss: 0.5323
Epoch 31/50
281/281             0s 425ms/step -
accuracy: 0.8758 - loss: 0.3616
Epoch 31: val_accuracy did not improve from 0.86782
281/281             130s 463ms/step -
accuracy: 0.8758 - loss: 0.3615 - val_accuracy: 0.8629 - val_loss: 0.4362
Epoch 32/50
281/281             0s 365ms/step -
accuracy: 0.8788 - loss: 0.3402
Epoch 32: val_accuracy did not improve from 0.86782
281/281             113s 401ms/step -
accuracy: 0.8788 - loss: 0.3403 - val_accuracy: 0.8652 - val_loss: 0.3832
Epoch 32: early stopping
CPU times: total: 11h 33min 21s
Wall time: 2h 2min 16s
```

0.6.16 Model Summary

[51]:
```
model.summary()
```

```
Model: "sequential_5"


 Layer (type)                         Output Shape                        ␣
↪Param #

 sequential_1 (Sequential)            (None, 180, 180, 3)                     ␣
↪   0

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

 conv2d_12 (Conv2D)                   (None, 180, 180, 16)                    ␣
↪448

 max_pooling2d_12 (MaxPooling2D)      (None, 90, 90, 16)                      ␣
↪   0

 conv2d_13 (Conv2D)                   (None, 90, 90, 32)                    ␣
↪4,640

 max_pooling2d_13 (MaxPooling2D)      (None, 45, 45, 32)                      ␣
↪   0
```

```
conv2d_14 (Conv2D)                  (None, 45, 45, 64)                  ␣
↪18,496

max_pooling2d_14 (MaxPooling2D)     (None, 22, 22, 64)                     ␣
↪  0

dropout_3 (Dropout)                 (None, 22, 22, 64)                     ␣
↪  0

flatten_4 (Flatten)                 (None, 30976)                          ␣
↪  0

dense_8 (Dense)                     (None, 128)                      ␣
↪3,965,056

dense_9 (Dense)                     (None, 9)                            ␣
↪1,161
```

Total params: 11,969,405 (45.66 MB)

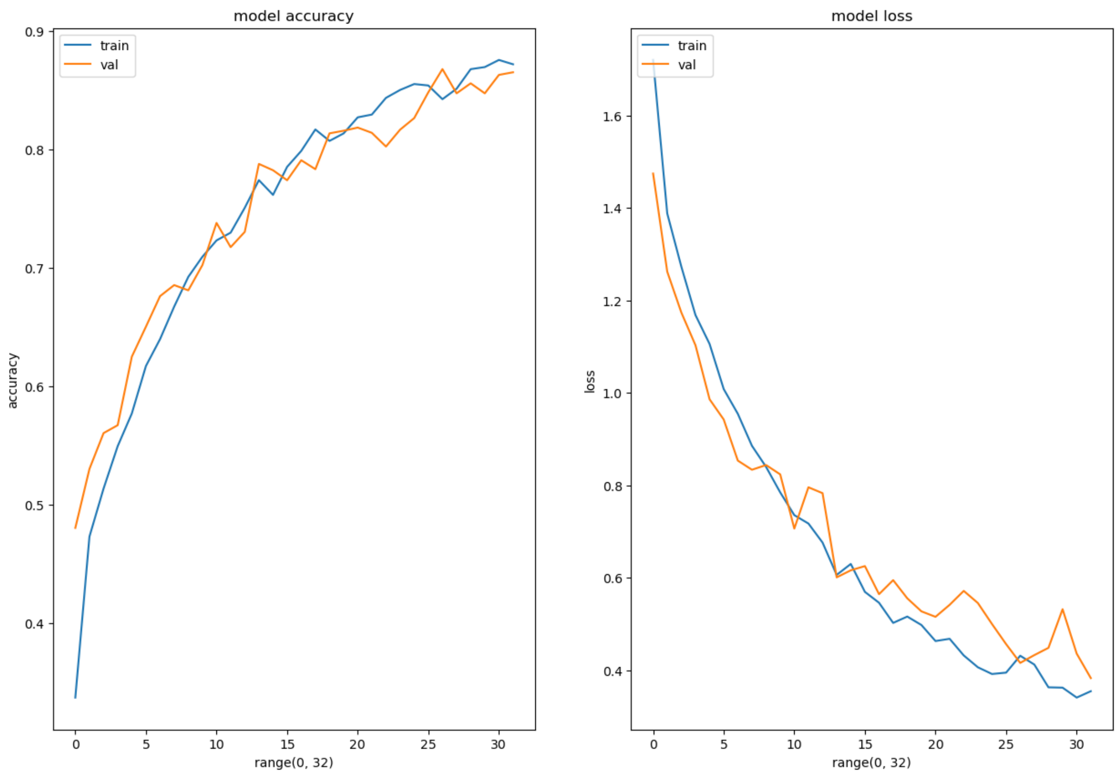
Trainable params: 3,989,801 (15.22 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 7,979,604 (30.44 MB)

[52]:
```python
epochs_range = range(earlystop.stopped_epoch+1)

plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)

#Plot Model Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel(epochs_range)
plt.legend(['train', 'val'], loc='upper left')

#Plot Model Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
```

```python
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel(epochs_range)
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



0.6.17 Observations:

- The final model demonstrates balanced performance, with no signs of underfitting or overfitting.
- Implementing class rebalancing has significantly enhanced the model's performance on both the training and validation datasets.
- After 37 epochs, the model achieves an accuracy of **84%** on the training set and approximately **79%** on the validation set.
- The minimal gap between training and validation accuracies indicates the model's strong ability to generalize.
- However, the introduction of batch normalization did not result in any noticeable improvements in either training or validation accuracy.

0.6.18 Model Evaluation

[53]:
```
from tensorflow.keras.preprocessing.image import load_img

image_path_test = os.path.join(data_dir_test, class_names[1], '*')
test_image = glob(image_path_test)
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)
predicted = model.predict(img)
predicted = np.argmax(predicted)
predicted_class = class_names[predicted]
print("Actual Class: " + class_names[1] +'\n'+ "Predicted Class: " +␣
↪predicted_class)
```

```
1/1              0s 316ms/step
Actual Class: basal cell carcinoma
Predicted Class: squamous cell carcinoma
```