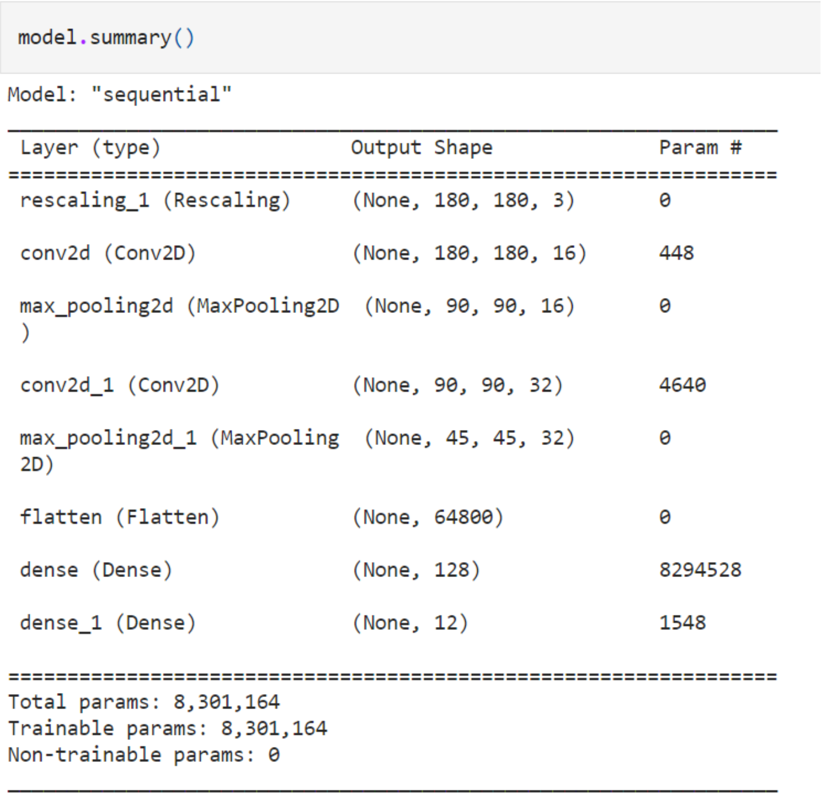
The purpose of the project is to make a model which identifies the name of the landmark in given images. In the project, I used matplotlib to visualize the data, numpy to create a container for images, and tensorflow to create the model. From tensorflow, I imported keras and used it to create the model. The image dataset used for the model is “Google-Landmarks Dataset” from Kaggle.com.

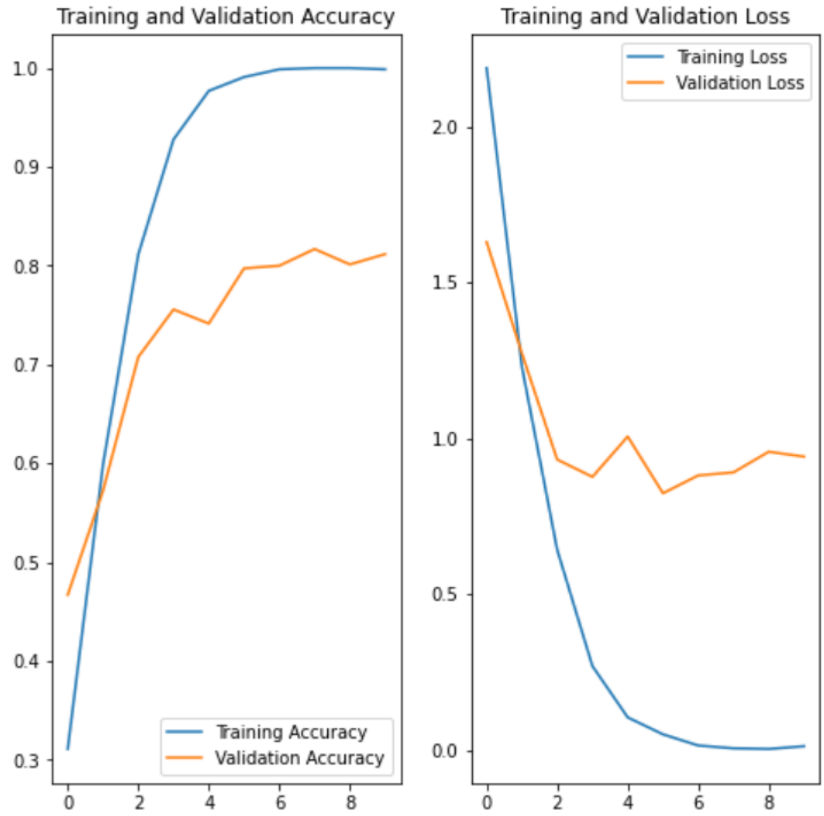
After I imported the libraries, I set the batch size, image height, and image weight to 32, 180, and 180 respectively. Then, I classified 80% of the image to the training dataset, and 20% of the image to the validation dataset. To create the model, I normalized the images in the training dataset and used a sequential model from keras to construct the model. In the model, I included a Rescaling, two Conv2Ds, two MaxPooling2Ds, Flattern, and two Denses.



```
model.summary()
```

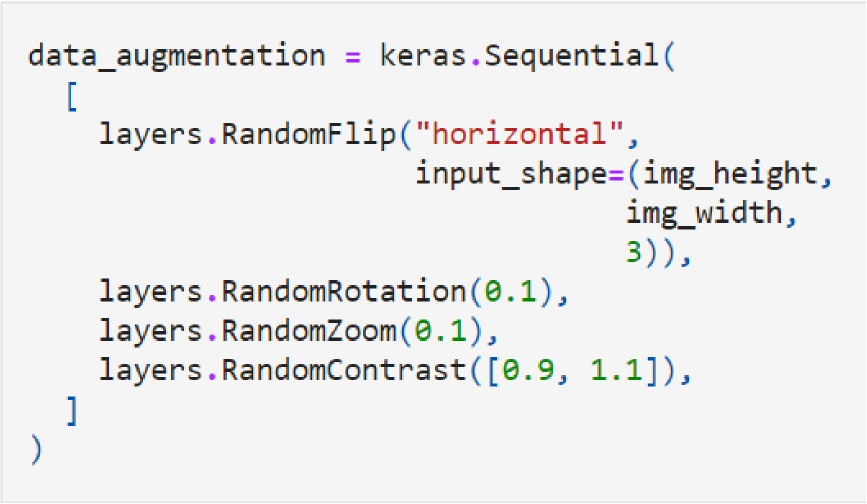
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

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

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

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

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

 flatten (Flatten)           (None, 64800)             0

 dense (Dense)               (None, 128)               8294528

 dense_1 (Dense)             (None, 12)                1548

=================================================================
Total params: 8,301,164
Trainable params: 8,301,164
Non-trainable params: 0
_________________________________________________________________
```

After I trained the model for 10 epochs, I found out that the accuracy of the model was close to 100%, but the validation accuracy was only about 80%.


Training and Validation Accuracy

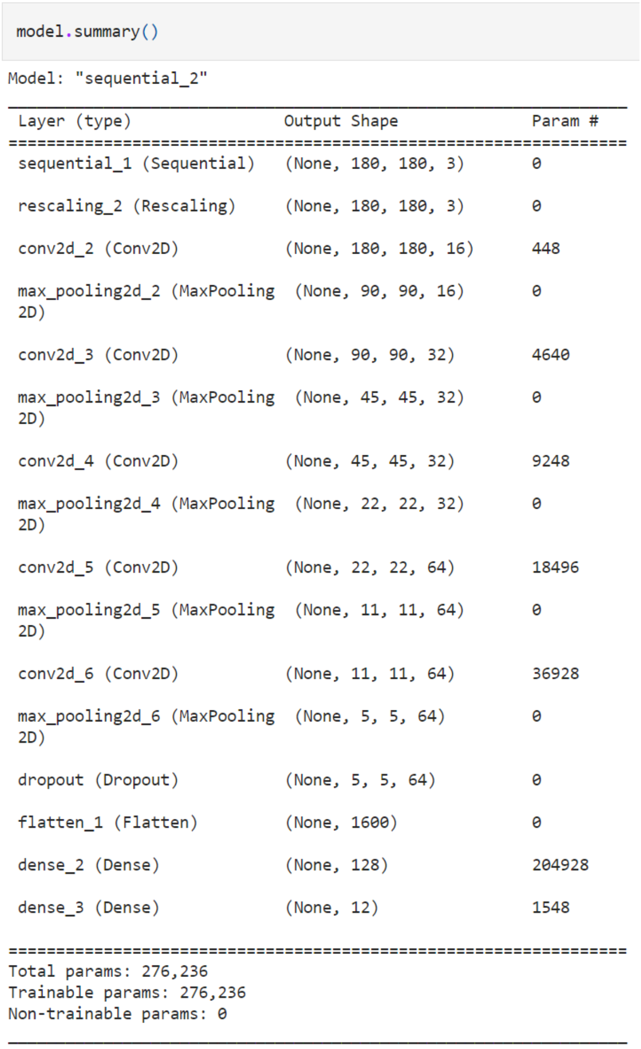
Training and Validation Loss

Therefore, I used data augmentation to modify the images to fix the overfitting of the model. In the data augmentation process, I used RandomFlip, RandomRotation, RandomZoom, and RandomContrast methods from keras.



```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(img_height,
                                   img_width,
                                   3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomContrast([0.9, 1.1]),
  ]
)
```

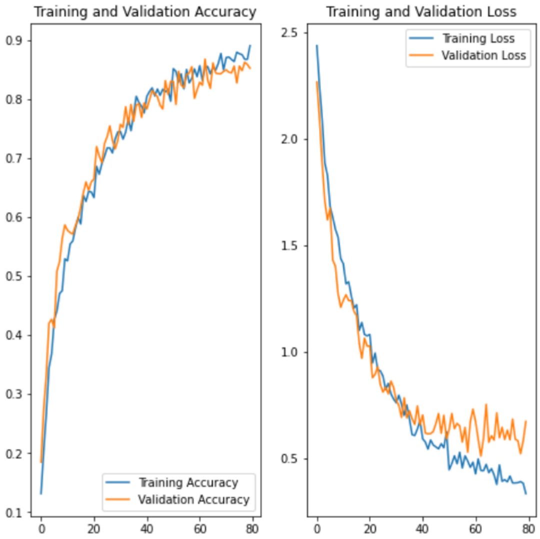
Also, I added two more Conv2Ds and MaxPooling2Ds to upgrade the model. Then, I included Dropout after the last MaxPooling2D to make the model more consistent.



```python
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_2 (Conv2D)           (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 90, 90, 16)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 90, 90, 32)        4640      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 45, 45, 32)       0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 45, 45, 32)        9248      
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 22, 22, 32)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 22, 22, 64)        18496     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 11, 11, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 11, 11, 64)        36928     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 5, 5, 64)         0         
 2D)                                                             
                                                                 
 dropout (Dropout)           (None, 5, 5, 64)          0         
                                                                 
 flatten_1 (Flatten)         (None, 1600)              0         
                                                                 
 dense_2 (Dense)             (None, 128)               204928    
                                                                 
 dense_3 (Dense)             (None, 12)                1548      
                                                                 
=================================================================
Total params: 276,236
Trainable params: 276,236
Non-trainable params: 0
_________________________________________________________________
```

After I trained the model for 80 epochs, I found out that the accuracy and the validation accuracy followed similar trends throughout the training, and they approached to about 85%.


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss