Deep Learning Project Report

Braeden King

30051656

Introduction:

In this project, a dataset of over 1000 outdoor images of different weather were used to train and test a convolutional neural network to classify them as cloudy, rainy, shine or sunrise. To classify these images, python was used along with a few libraries, mainly TensorFlow to build the model, OpenCV for image preprocessing, NumPy for data modification, and matplotlib for the visualization of data.

Preprocessing:

All preprocessing in this project was done with OpenCV. First the image was loaded using imread(). Then it was resized to be 100x100 pixels. These dimensions allowed for an image that was still recognizable while not being so large that the model took a long time to compute. After resizing, the colour was converted from OpenCV's BGR to RGB for the purpose of displaying it. The image pixels were then normalized to between 0 and 1 by dividing by 255.0 and were then appended to the training list with every fifth image being assigned to the test list. Lists for the labels of the images were then created along with one-hot encoded versions of these lists. 10 images from the training set were then displayed on screen to show the preprocessing that was performed.

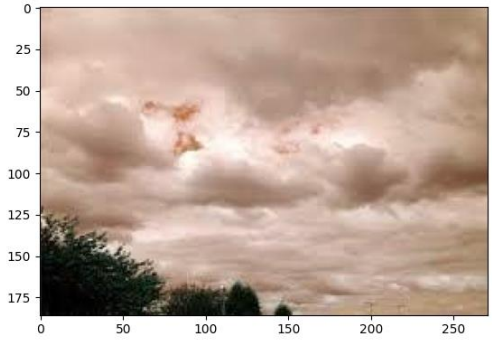

Figure 1.1 - Original Image

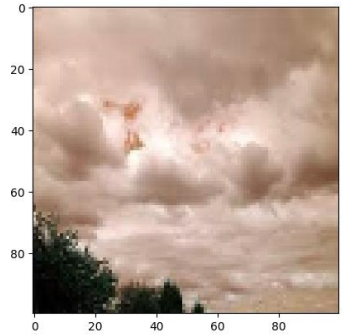

Figure 2.2 - Resized Image

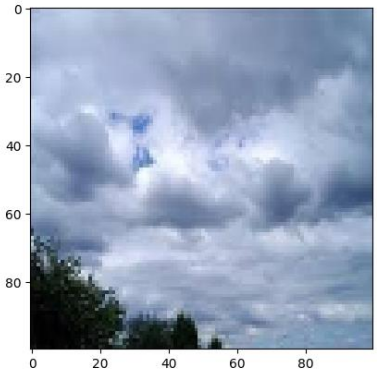

Figure 1.3 - Colour converted Image

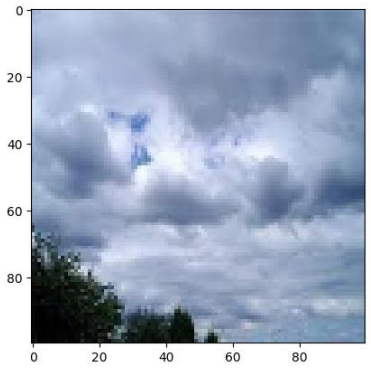

Figure 1.4 - Normalized Image

Approach:

To build the model, TensorFlow was used. This model was built Sequentially instead of Functionally and started off a 32 neuron 2D Convolution layer with a 3x3 kernel and a ReLU activation function that took a 100x100 RGB image. After this layer there were two sets of pooling and convolution layers. Both pooled with a 2x2 area then outputted to 2D convolution layers with 3x3 kernels, 64 neurons and a ReLU activation function. After these layers, the outputs are flattened and sent through a 64-neuron dense layer with ReLU activation function and then through the 4-neuron dense layer with softmax activation function.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 98, 98, 32)        896

 max_pooling2d (MaxPooling2D  (None, 49, 49, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 47, 47, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 21, 21, 64)        36928

 flatten (Flatten)           (None, 28224)             0

 dense (Dense)               (None, 64)                1806400

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 1,862,980
Trainable params: 1,862,980
Non-trainable params: 0
_________________________________________________________________
```

Figure 2 - Model Summary

This model went through many different iterations before arriving at this. One of the first approaches to obtain a higher accuracy value was to add more pooling and convolution layers in the middle of the model. This led to a longer time training the model along with a lower accuracy value. The next approach was to use larger images as input into the model, so an image size of 300x300 pixels was chosen. This led to a much longer training time up to 5 minutes from around 45 seconds and only saw a marginal increase of around 1-2% in accuracy. Due to the much longer training time with such little accuracy increase, 100x100 pixel images were chosen as the input. These parameters gave a model that when trained yielded a 92% val_accuracy level with val_loss of 0.3181 when trained over 10 epochs.

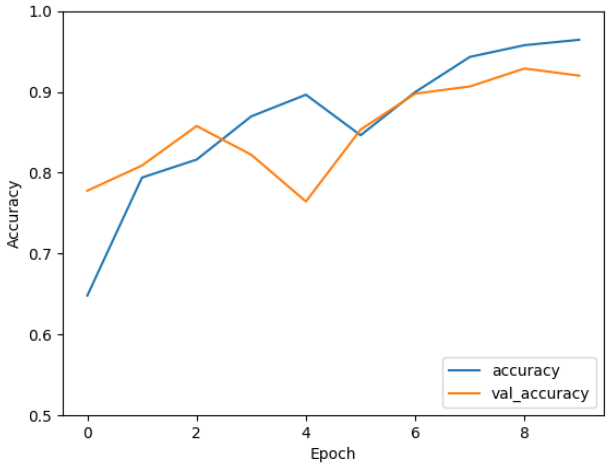

Figure 3 - Accuracy vs. Epochs

Results:

The model created performed quite well on the test set of images, classifying them with around 92% accuracy. This meant that of the 225 test images, 207 were correctly classified (true positives), while 18 were incorrectly classified (false positives). The class that was most often correctly identified were sunrise images, while cloudy images were the most likely to be misclassified.

```
Dataset Confusion Matrix:
[[52.  3.  5.  0.]
 [ 1. 41.  1.  0.]
 [ 3.  2. 45.  1.]
 [ 1.  0.  1. 69.]]
```

Figure 4.1 - Confusion Matrix of dataset test images

When testing images from my own collection of weather photos, there was a large decrease in the model's ability to classify images correctly. It saw a 32% decrease in accuracy from the dataset test images leading to 60% accuracy. This meant that it correctly classified 6 of the 10 images. The most likely to be misclassified was the cloudy images, which were misclassified as rainy images 3 times.

```
Own Images Confusion Matrix:
[[4. 3. 0. 0.]
 [0. 0. 0. 0.]
 [0. 1. 1. 0.]
 [0. 0. 0. 1.]]
```

Figure 4.2 - Confusion Matrix of own test images

Discussion:

When apply my own test images to my model, there was a large decrease in its accuracy when compared to the dataset test images. There are two reasons that this occurred, both stemming from the images chosen. The first problem was the focus of the photos. While the images in the dataset had a focus on the weather present in the photos, the photos chosen by me were mostly images I had taken in the past that did not have a focus on the weather, but something else in the foreground. This led to the weather in the photo being less distinguishable from the foreground. The second issue with these images is a lack of distinction in the weather in the photos. Many of the photos chosen did not have distinct weather that aligned with a single class but had weather that fit multiple classes. This can be seen in images that had the sun shining through clouds, which could be classified as weather sunshine or cloudy.

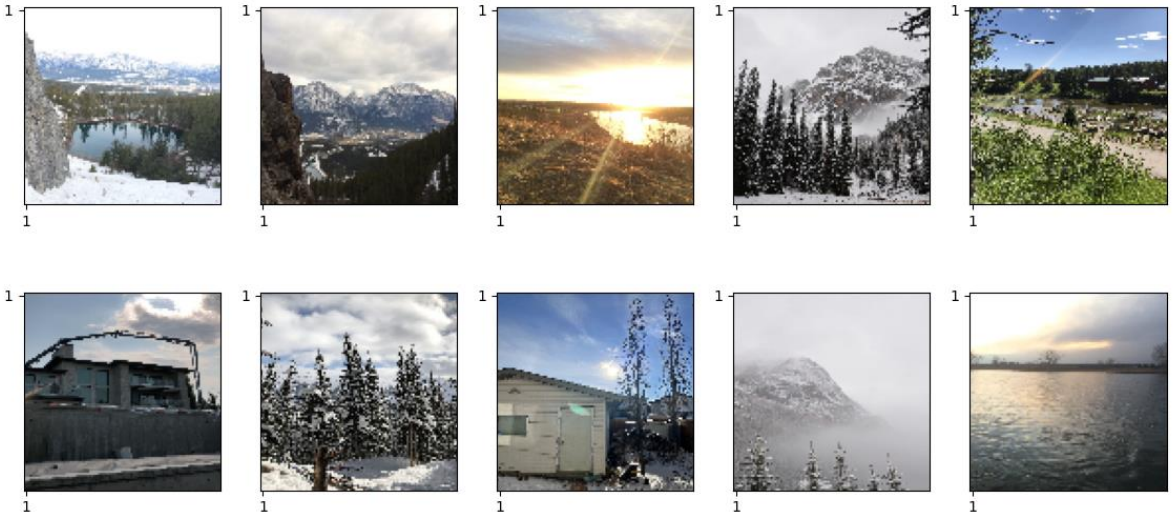

Figure 6 - Test images taken by me

```
Actual labels:
[0, 0, 3, 0, 2, 0, 0, 2, 0, 0]
Predicted labels:
[1, 0, 3, 1, 1, 0, 1, 2, 0, 0]
```

Figure 5 - Actual and Predicted images labels
0 – Cloudy, 1 – Rainy, 2 – Shine, 3 – Sunrise

Conclusions:

The model created performed quite well with 92% accuracy on the dataset. It correctly classified images with distinct weather in them, and only misclassified 18 images on the dataset. The model showed its limitation when applying test images that did not have distinct weather in them as well as images that had noise in the foreground. The model struggled most with classifying cloudy images, either misclassifying them as rainy or shine. Overall, the model did quite well, correctly classifying 207/225 dataset images and 6/10 of my own images.