Python ML Project Part 2

Team 2

**Resizing the Data**

The TensorFlow library requires for all image data to be represented in a NumPy array, and for all images to have the same dimension to maximise training accuracy. We resized all the images using CV2 to 224 x 224 pixels.

**Training the model: First Run**

We first split the dataset from the train folder into two sets of data – 90% of the data was used to train the model, whereas the remaining 10% was used to validate the model after each epoch. The 60 images from the test folder were used to evaluate the final model and was not used for validation.

We first ran the model with the following layers:

1. *Convolution2D; 32 convolution layers, ReLU Activation*
2. *Pooling;*
3. *Convolution2D; 32 convolution layers, ReLU Activation*
4. *Pooling;*
5. *Convolution2D; 64 convolution layers, ReLU Activation*
6. *Pooling;*
7. *Flatten*
8. *Dense; 64 units, ReLU Activation*
9. *Dropout;*
10. *Dense; 4 units (same as the number of classes), Softmax Activation*

Upon evaluating our final model with the test images, we obtained an accuracy score of 0.83 and a loss of 1.40.

**Methods taken to improve model accuracy**

**Method 1: Data Augmentation**

Machine learning requires a large dataset to maximise the accuracy and minimise the loss of the classification model. Our training dataset comprising 240 images is small and could thus have resulted in the low accuracy score obtained above. To rectify this problem, we used data augmentation to increase the variety and size of our dataset. The current dataset was randomly rotated, mirrored or flipped and subsequently fitted into the same model to train the model.

We evaluated the model with the test images again, and recorded a greater accuracy score of 0.85. The loss also decreased to 1.01. Data augmentation was indeed able to improve the accuracy of our model and reduce the loss of data, but the final model was still not optimal.

**Method 2: Resizing the Data**

When we were training the 2 models previously, we noticed that the training accuracy of the model was close to 1.00, and the training loss was also less than 0.1. The lower accuracy and greater loss when evaluating the model with the test data was thus indicative of model overfitting. In order to reduce overfitting, we decreased the size of the original training and testing images to 180 x 180 pixels (without the previous data augmentation) and subsequently evaluated whether a smaller image size will impact the accuracy of our model.

Like before, we split the dataset from the train folder into two sets of data – 90% for training and 10% for validation. We trained the using the model with the same layers and observed an accuracy score of 0.9 and a loss score of 1.08 when evaluating the final model with the test images. From these observations, we can infer that decreasing the image size led to a greater increase in the accuracy as compared to the data augmentation method, whereas the decrease in loss in this method was similar to the decrease in loss in method 1.

After studying the image dataset in greater detail, we noticed that some of the images are smaller than 224 x 224 pixels. When we previously resized and magnified these smaller images to 224 x 224 pixels, we could have increased the noise and reduced the resolution of these smaller images. This might have reduced our data quality, causing our model to be less accurate in the previously.

**Method 3: Resizing the Data and Data Augmentation**

We combined both data augmentation and resizing the data to a smaller image size and observed how this affects the model trained.

We applied data augmentation to the dataset which contains images resized to 180 x 180 pixels and train our model again using the same layers as before. While the accuracy score remained at 0.9, the loss score has decreased significantly from 1.08 to 0.53. The current model is thus able to predict the correct labels with a higher certainty. This is the result of both reducing the noise and increasing the size of the training dataset, thereby allowing the model to better adjust the weights based on the important features in the image. The lower loss and high accuracy from the testing data also indicate that there was less overfitting when training the model.

Please refer to the Jupyter Notebook: **Part II (Data Augmentation & Image Resizing)** for the codes used, the accuracy score diagram and the loss diagram for all the previous sections.

**Method 4: Adding an additional convolution layer**

We attempted to increase the accuracy by adding another convolution layer into the current model. We pre-processed and resized the images to 224 x 224 pixels, and split the data using the same methods described above. We also applied data augmentation on the training dataset as detailed in Method 2. Upon evaluating the final model with the test images, we obtained a high accuracy score of 0.88. However, the loss score increased drastically to 260. This showed that by increasing the number of convolution layers increased the overfitting of the model.

Please refer to **testfilePY.py** for the code used, accuracy score diagram and the loss diagram.