End-to-End Optimisation in TensorFlow
Assignment 3

This assignment aims to understand the learning outcomes of TensorFlow data processing pipelines and understand how GPUs can accelerate data processing. Write data processing pipelines that exploit TensorFlow and have the knowledge to understand how to develop GPU-accelerated data processing pipelines using the TensorFlow and RAPIDS framework. The three methods **CNN**, **Image Pre-Processing**, and **DA** by which data processing is performed in this assignment are explained below.

Convolutional Neural Network (CNN)

According to a report a deep learning algorithm that takes an input image, assigns importance (learnable weights and biases) to various aspects and objects in the image and can differentiate one from the other is called a Convolutional Neural Network. The pre-processing required in a CNN is much lower as compared to the rest of the classification algorithms. While in basic methods filters are hand-engineered, with sufficient training, CNN has the ability to learn these filters.

Image Pre-Processing

According to a report Image pre-processing is the ability to improve the quality of an image so that we can analyse it in a better way. Using pre-processing we can minimize the undesired distortion and improve some features which are necessary for the specific applications we are working for. These features might be different for different applications.

Data Augmentation. (DA)

According to a report a machine learning technique that can be used to artificially expand the size of a training set by generating modified data from the existing data is called Data Augmentation. It is said that using DA is a good practice if you want to prevent overfitting. A DA can also improve the performance of the model by augmenting the data we already have, which means that DA is also good for enhancing the model's performance.

Comparing Exercise 1 and Exercise 2.

Training the network and plotting the resulting loss functions and metrics. This is a pre-defined code by Ashley in exercise 1 and in exercise 2 I have created the pipeline from the scratch. The image size used is **(128,128), Batch_size = (64), with epochs = 20** in exercise 1 and the same is different in exercise 2.

Both the exercises have loaded the cats and dog datasets using **tfds.load** which loads the name datasets in double-quotes. After loading the dataset, it is reserved as **10% validation** and **10% test** and the rest is reserved for training. By setting the datasets with the augmentation and resizing settings, some of the images in the datasets were corrupted and skipped.
In my version of the pipeline function along with **image resizing** and **rescale**, I have also used the functions for **augmenting** the data.
For pre-processing along with **train/test, Autotune** has been used for better performance of the results.

The shape of the image after pre-processing is a tensor of the shape **(64, 224, 224, 3)**. This is a batch of 64 images of shape **224 x 224 x 3**. (The 3 represents the color channels RGB). The 64 represents the corresponding labels to the 64 images.
After successfully loading the dataset and plotting visualization by test/split, functions for model history and visualization for the correct prediction and wrong prediction were defined to use in the later stages of the pre-trained model for training and predicting on the datasets.
For the pre-trained model for training and predicting I have used **Transfer Learning (MobileNet).**

Transfer Learning (MobileNet)

Transfer Learning

In simple terms, Transfer Learning is the process of using a pre-trained model that has already been trained on a dataset for training and predicting on a new defined dataset. A pre-trained model is a model that has been previously trained on a large dataset, typically on a large-scale image classification process.

Transfer Learning with MobileNet

MobileNet model was developed by Google, which is a pre-trained model on the ImageNet datasets with 1.4M images and 1000 classes of web images.

The MobileNet model has a dense layer activated by the SoftMax activation function for interpreting the result as a probability distribution. This model has a Total parameter: 3,230,914, Trainable parameters: 3,209,026, and Non-trainable parameters: 21,888.

The history model.fit has been executed with epoch = 5 with the first epoch having a loss of 0.2861 and an accuracy of 0.8726, and the final epoch (epoch 5) has an improved loss of 0.1573 and an improved accuracy of 0.9359.

Model History Plotting

The graph for Training Accuracy vs Epochs and Training/Validation Loss vs Epochs can be seen in the output in the notebook. The first graph Training Accuracy vs Epochs shows that the training accuracy has increased at 1 epoch and till it reaches epoch 4 it gets increasing more because the accuracy is less while the validation accuracy started to decrease at epoch number 3 because the accuracy increases because we are training the model.

Actual predicted results and wrong results

The predicted result has generated 24 images in total of cats and dogs with 4 rows and 6 columns. The model has made a wrong prediction of two actual dogs in the first row and has identified them as a cat.

- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- https://medium.com/spidernitt/image-preprocessing-why-is-it-necessary-8895b8b08c1d

- https://analyticsindiamag.com/a-practical-guide-to-implement-transfer-learning-in-tensorflow/

- https://analyticsindiamag.com/a-practical-guide-to-implement-transfer-learning-in-tensorflow/#:~:text=Transfer%20learning%20is%20simply%20the,%2Dscale%20image%2Dclassification%20task.