**Coursework report**

**Task 1:**

From the cloud speed test we gathered that reading f rom the pre-process TFRecord files is much faster then reading the data on it’s own terms and so know we are going to going to explore, understand and quantify the effect of this when doing speed implementation.

We implement this in batch size, numbers and repetitions from example in this case we took from between 1 to10 and so once that is done we then added value of x axis and y axis value so it obtains a throughput image per second.

While this done we moved on to linear regression so it can use speed and parameters for two separate datasets.

**Task 2:**

Now we are using spark to analyse the performance of the speed test and so do this we are going to use multiple parameters in parallel with spark- therefore that would a cloud, because in big data it’s important to utilise external source to test speed. In this task Dataproc was used to run and enabled. Dataproc is IDE plugins that are used to interact with Google APis, and so a function was used to to create a cluster and manage the google clusters and dataproc. Also another term was used to perform an action was the use of Spark RDDs in a distributed way and so the parameter we mention from the offset will be utilised here except the repetition parameter. The RDD was then created to run average speed and to collect results for each parameter.

The two main aspects to improve efficiency and productive of clusters is to apply the Caching and parallelisation branches technique. Caching can give you predictive time and reduce wasting it as it allows you to access the RDD more the once as for example for speed we can use the function rdd.cache() to speed the test and so because of the caching is existing result is vital.

In terms of parallelisation branches this is applied when you instruct spark to map the computation to additional nodes. Therefore this another vital speed trick to be used when you have more than 2 nodes to speed up using cloud based CPUs to implement the computation in more branches. Here is the result of the test figure below.

Chart, scatter chart

Description automatically generated

Above visual show results of speed test done, the x axis is the parameter given and y axis is the throughput in images per second, this was done by using linear regression each parameter for the two cases (reading from image files/reading TFRecord files.

Chart, line chart

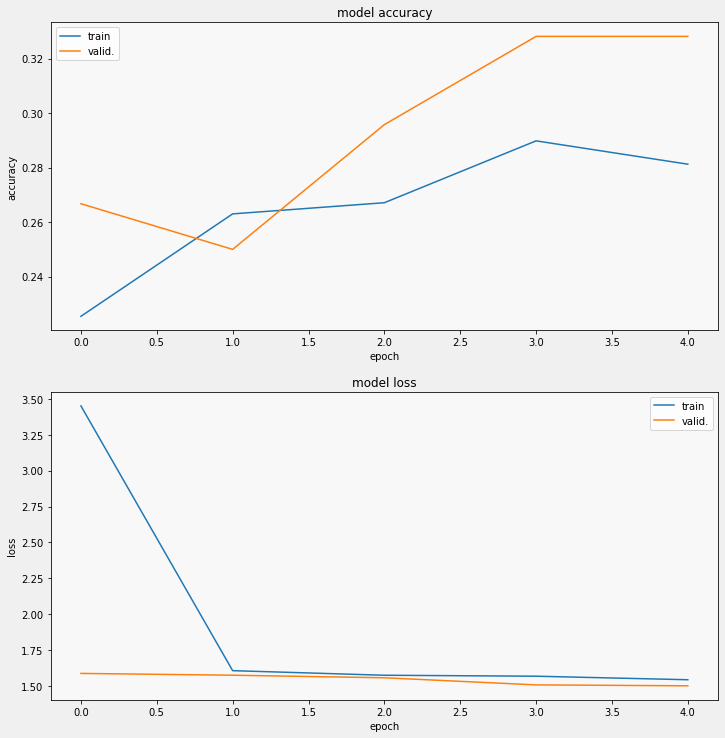
Description automatically generated

Here we can understand that the latency impacts the images directly in the lower scale where the decoded wouldn’t be too aggressive in terms of the impact. From the analysis we can also inform ourselves that the using a large scale machine learning can increase the image throuput through using more GPus application and higher parameters and so the machine would not be too slow and hence the performance wouldn’t be effected. Below is the linear regression speed test performance.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| decoded | slope | intercept | R value | P value | gradient |
| bs | 3.8888 | 348.34232 | 0.4442 | 7.312121e-01 | 0.232424 |
| bn | 49.21212 | 266.4223232 | 0.23222 | 4.231212e-09 | 8.12121 |
| ds | 0.23222 | 532.1212 | 0.7889 | 3.223232e-02 | 0.19199 |
| rep | 5.32232 | 883.120843 | 0.075413 | 6.312121e-03 | 76.21212 |

**Task 3:**

From the graph we can find that the CPU utilisation is far lower then I/O and network is around the same. The reason why network is much higher is because of computation used with spark GPU and so a machine on its own would be a down grade those with upgrades of cloud tools and other technologies such as spark. In this section we also utlised a model training . Although in the code it doesn’t show the result.



Chart, histogram

Description automatically generated

**Task 4:**

The methodology used for testing distrubatuion and cluster type was in this order - multiworkermirrored, mirrored, oneDevice and the rest of CPUs 8 and 32 datas (standard and complex).

In our experiment we implemented various approaches to the performance of complex clusters and standard ones and the above and to see the speed. In the notebook we tired from lower rating and size so we can see the performance for example we started from 32 batch size going up to 128 and so in terms of memory that would around 15gb to 150 gb and so the other method was used to see the GPU to see the differences in the model. Overall in general I think to have use a complex experimented or datasets we should need a external resources such hard drive Accelerometer gpu or spark cloud to manage a huge parameters with a good speed and accuracy.

Chart, line chart

Description automatically generated

**Task 5:**

From the papers [1] and [2] we can explore various of parameters and cloud methods to have a best contextual result.

We can learn that by increasing the batch size of a model we can have more robust model other then moving into cloud according to Smith . However we can also understand there is advantages and disadvantages in using cloud or more batches size into machine for training. In the papers we can learn that sometimes even if we have the highest speed in training the data however we might lack in other factors such as accuracy and performance wise, therefore this is critical to understand the benefit and loss.

In terms of the strategies they are three ways to implement this – ie batch processing, real time processing and stream processing. To apply the batch processing as mention before there is shortfall to it high data to be processed as it can be become problematic in a long term and so the best way to use this strategy is obtain a high portable CPu. For the real time and stream processing I think they are closely linked strategy and that strategy is based on computer network algorthims, TCp, Ip fast network bandweidth and larger memory storage would help in the future and now and of course its still important to have a good cpu to store those data and algorithms as the continuous processing require this flexibility.

Chart, line chart, histogram

Description automatically generated

**References**

[1] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming

Zhang. Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In NSDI,

2017.

[2] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don’t decay the learning rate, increase the batch

size. ArXiv, abs/1711.00489, 2018.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated