1. Choose two different DNN models available for your chosen dataset. a. AlexNet, ResNet, and MobileNet for ImageNet dataset b. Use any model for Word2Vec that you can find for your assigned DL framework. c. Linear Model, Linear Model with Crosses, and DNN for Criteo Click Logs.

For this experiment, we chose the Word2Vec 1- billion words dataset.

1. (50 points) Perform the above experiments on a single node (CPU and GPU) using the DL framework assigned to your team. For the single node CPU run, you can vary the number of cores and number of threads in the application. For GPU runs, there is no need to vary the number of threads, instead, use different batch sizes to find out the best performing batch size.

For the training model, we chose the gated convolutional networks [1]: a major contribution of the model is that it was a new attempt to use a non-recurrent approach to attain near performance as recurrent neural networks. The model is based on convolutional networks with a gating mechanism.

We conducted corresponding CPU experiments on Owen clusters and GPU experiments on Pitzer clusters for queuing time and computation time purposes. For the experiment of both CPU and GPU, we chose three values for the batch size: 32, 64, and 80. For the number of threads, we chose 14 and 28 for CPU training. We will run 5 epoch for this model.

1. (5 points) Mention the scale and size of your dataset and explain why you chose it?

The scale of Word2Vec is rather large in terms of the quantity of the words; however, its overall size is the smallest of all three datasets. We wanted to choose the ImageNet datasets since we all have a background on the image processing with DNN, however, the size of the dataset is so large that it takes so long to download and unzip, considering both the instable speed of the network in the environment and the difficulties to share a large dataset between different accounts in the OSC environment. As a result, we chose Word2Vec for its abundancy and relative small volume.

1. (5 points) What is the best DNN architecture for your chosen dataset in terms of “accuracy” and in terms of “training time”? Describe if you find a suitable tradeoff between these two metrics.

The model I chose is special for its innovative nature: it use convolutional network to train text dataset. However, it also shows non-trivial performance. For the training time, the model converged very fast and the test accuracy stabilized around 42% to 43% for most of the time. Table 1 shows the final accuracy after 5 epoch of each experiment under different hyperparameters and Table 2 shows the average training time of all 5 epoch for CPU experiments.

|  |  |  |  |
| --- | --- | --- | --- |
|  | CPU with 7 threads | CPU with 14 threads | CPU with 28 threads |
| Batch size = 32 | 0.4018 | 0.4021 | 0.0898 |
| Batch size = 64 | 0.4297 | 0.4303 | 0.4295 |
| Batch size = 80 | 0.4341 | 0.4303 | 0.4353 |

Table 1 Final accuracy among all epochs for each CPU experiment

|  |  |  |  |
| --- | --- | --- | --- |
|  | CPU with 7 threads | CPU with 14 threads | CPU with 28 threads |
| Batch size = 32 | 4109.6 | 3789.2 | 5235 |
| Batch size = 64 | 3683.4 | 2178.6 | 3185.2 |
| Batch size = 80 | 2044.4 | 1843.6 | 2038.6 |

Table 2 Average epoch time among all epochs for each CPU experiment

|  |  |  |
| --- | --- | --- |
|  | accuracy | training time |
| Batch size = 32 | 0.1077 | 538 |
| Batch size = 64 | 0.4328 | 291 |
| Batch size = 80 | 0.4323 | 273 |

Table 3 Accuracy and training time of GPU experiments

1. (5 points) Is the GPU always better than CPU for training with larger dataset? Explain your observation.

In terms of training time, GPU shows an overwhelming advantage over CPU clusters, which can be 10 times faster than CPU. This shows the powerful lead of GPU. However, in terms of final accuracy, we can see very close accuracy for GPU and CPU; for some cases, CPU with 28 threads even outperforms GPU for batch size = 80.

A very interesting phenomenon is that with batch size = 32, CPU with more threads and GPU diverged and achieved much worse performance than CPU. This could be because the optimizer the model uses (Adadelta) is not suitable for small batch. Another possibility is because of the model per se: when running batch size = 1024 after epoch 40, although the loss is steadily decreasing, the model’s test accuracy decreases. This is a very clear signal of overfitting. However, again, we cannot be sure since we did not train the model for more than 100 hours like the paper.

1. (5 points) For the single-node CPU run, do you see any trend when varying the number of cores and number of threads in the application? Explain your observation.

The model shows a very interesting pattern: CPU with 14 cores shows better performance than both 28 cores and 7 cores. Larger amount of CPU cores can provide more computational power, however, due to larger communication overhead between different cores, the overall performance will be worse; for smaller amount of CPU cores, the computational power becomes the bottleneck. As a result, for this gated convolutional model, it is the best to choose medium amount, compared to too large or too few amount of CPU cores.

Reference:

[1] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 933–941.