Comparative Study of TensorFlow and PyTorch on Single and Distributed Systems

Jenna Cooley
1218701263

Venkata Naga Aditya Datta Chivukula
1227267290

Archit Agrawal
1229565261

Introduction

This paper investigates the comparative performance of TensorFlow and PyTorch, two dominant deep learning frameworks, in the domain of sentiment analysis, a cornerstone of Natural Language Processing (NLP). The central focus lies in evaluating the efficiency and scalability of each framework within two distinct computational environments: single-processor systems and distributed multiprocessor configurations. Sentiment analysis, the process of classifying textual data based on its emotional sentiment, can be computationally demanding. As NLP applications continue to evolve in complexity and scale, selecting an appropriate framework becomes paramount for maintaining efficiency and cost-effectiveness.

The significance of this comparative analysis stems from the widespread adoption of TensorFlow and PyTorch within the deep learning landscape. Both frameworks boast extensive user bases across academia and industry. The insights gleaned from this investigation can empower a broad spectrum of practitioners who rely on these frameworks for their NLP endeavors. By meticulously evaluating performance across single-processor and distributed multiprocessor setups, this study aims to illuminate the relative strengths and weaknesses of each framework in scenarios that necessitate scalability and flexibility. The findings hold the potential to significantly impact research and commercial applications, guiding developers towards more efficient resource allocation and expedited training times. Ultimately, this analysis seeks to deliver a comprehensive assessment that can contribute to the development of more efficacious and scalable solutions within sentiment analysis and related NLP domains.

Literature Survey

Sentiment Analysis [1] is an important application of NLP which plays a crucial role in social nets analysis [2] and survey polls [3] during elections. In the early days, methods like cbow and skip gram were used to generate embeddings [4] and later, deep learning approaches like RNN [5], LSTM [6], etc. were employed for the same task. With the rise of transformers [7], models like BERT have shown prominence [8] in industry standard benchmarks like sst2 from GLUE [9]. Though foundational models like LLMs like GPT, Llama, mistral, etc. prove dominance in

generative AI but do not produce SOTA on downstream NLP tasks where models like DeBERTa V3 fare the best [10]. Hence, training metrics and information of such models on downstream NLP tasks like sst2 proves to be important data to analyze. Now, one can choose different frameworks for analyzing but, two main important frameworks are TensorFlow [11] and PyTorch [12].

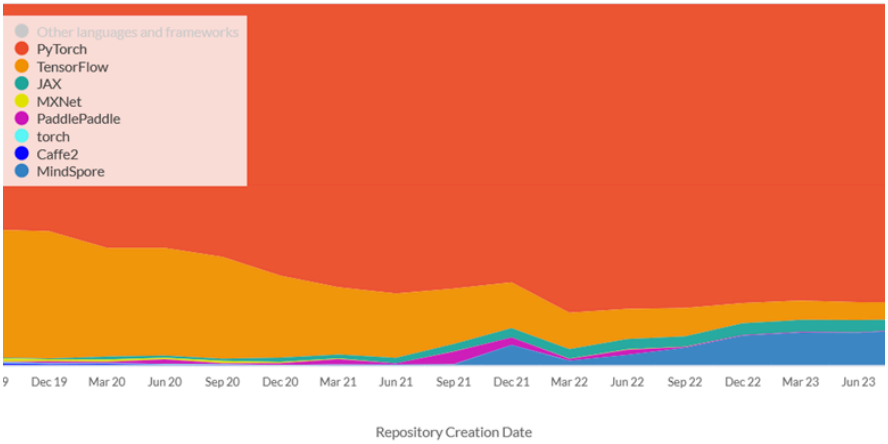

Figure 1: Framework adaption

As seen above in Figure 1, the most relevant frameworks to examine are PyTorch and TensorFlow. With increase in dataset size, distributed training is now the goto approach. Distributed Data Parallel [13] is one such approach that was introduced in early days but proves to be the best when the model size is not big enough for the processor but the dataset is huge. Dataset is divided into chunks and the model is copied on all the GPUs. Each model copy is trained simultaneously on different chunks of the data and the gradients are accumulated to get updated. This strategy is available in both TensorFlow and PyTorch and serves as a good comparison metric for the analysis.

Methods

The purpose of the project was to compare the performances of TensorFlow and PyTorch in single and distributed systems. The data will be retrieved from GLUE benchmark with proper tokenization applied using the default tokenizer for PyTorch and TensorFlow. Each model will be trained separately with a single GPU instance and multi GPU instance. The results will then be compared to evaluate which model performs the best. A standard comparison of model performance will be conducted, measuring accuracy by calculating the proportion of correct predictions over total number of predictions, employing metrics such as F1 score, precision and recall.

Setup

The project was set up to compare TensorFlow and PyTorch for sentiment analysis, using different types of computer setups. Here's how we organized everything:

1. Google Colab for single-processor tasks: We chose Google Colab because it's easy to use and Google Colab provides access to GPUs, which can significantly speed up the training of deep learning models. The standard Google Colab GPU provides NVIDIA T4 Tensor Core GPU with 12 GB of RAM and 12GB of GPU memory.

2. Kaggle for multi-processor tasks: We used Kaggle for tasks that needed more than one GPU because it provides 2 GPUs with the same configuration as Google Colab free of cost. This helped us see how each framework, TensorFlow and PyTorch, performs when scaled up and using more resources. Kaggle's setup was crucial for testing how well each framework could handle large-scale operations and have a common hardware configuration for comparing the model's performance.

3. Weights & Biases for monitoring: To keep track of everything like how well the models were performing and how much computer resources were consumed, we used a tool called Weights & Biases. It let us log detailed information and watch what was happening in real-time, which was important for comparing TensorFlow and PyTorch in different situations.

By using Google Colab, Kaggle, and Weights & Biases, we created a consistent and reliable way to study how TensorFlow and PyTorch work in both single and distributed systems. This setup helped us thoroughly test and compare the performance, scalability, and efficiency of both frameworks in sentiment analysis under different configurations.

Dataset

This study employs the Stanford Sentiment Treebank (SST-2) dataset, a portion of the GLUE benchmark commonly used for sentiment analysis tasks. SST-2 consists of labeled sentences from movie reviews, with each sentence assigned a sentiment class: positive, negative, or neutral. utilized this dataset to train and evaluate sentiment analysis models within the TensorFlow and PyTorch frameworks.

Results

Metrics Comparison	PyTorch		TensorFlow	
	1 GPU	2 GPU	1 GPU	2 GPU
Accuracy	92.08%	92.43%	91.51%	92.43%
F1 Score	92.34%	92.70%	91.57%	92.66%
Precision	91.03%	91.98%	92.62%	91.44%
Recall	93.69%	94.37%	90.54%	93.91%

Table 1: Metrics Comparison

Time Comparison	PyTorch		TensorFlow	
	1 GPU	2 GPU	1 GPU	2 GPU
Average Epoch Runtime(seconds)	560	610	980	1200

Table 2: Time performance comparison

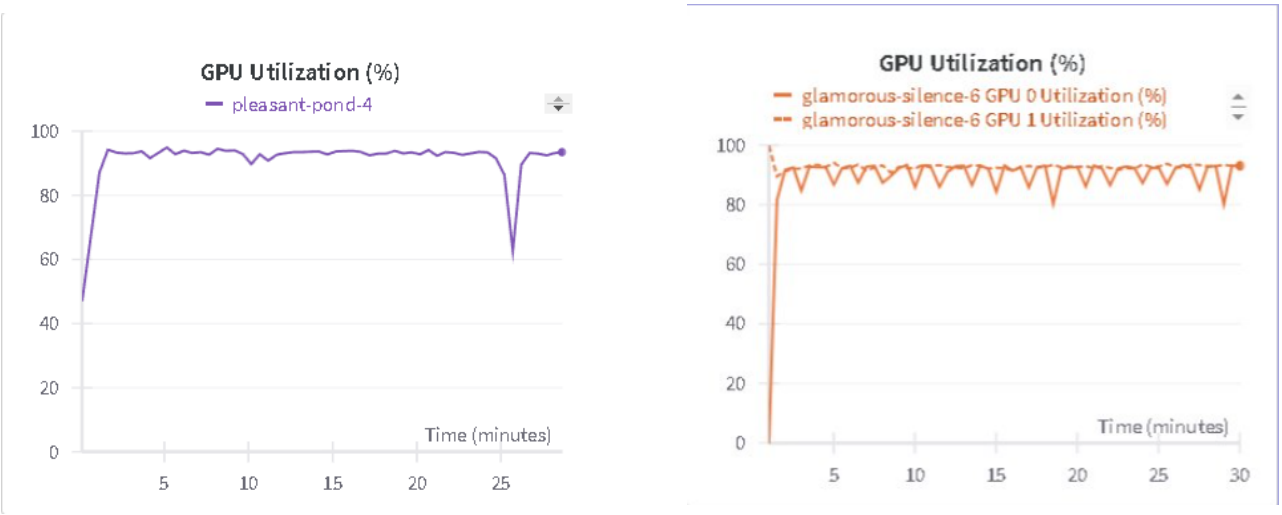

Figure 2: Pytorch GPU utilization comparison in 1 GPU(left) and 2 GPU(right)

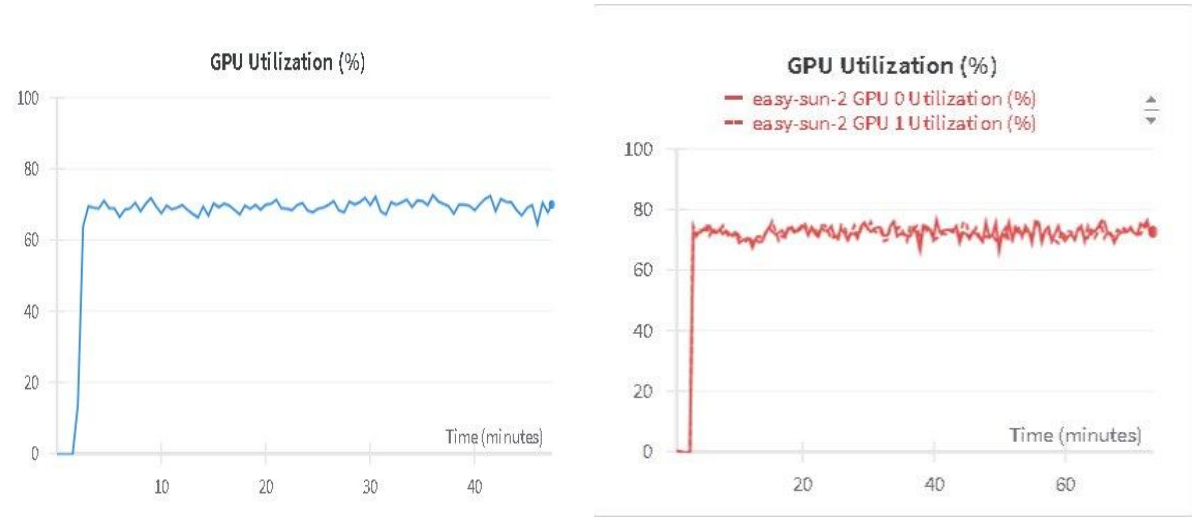

Figure 2: TensorFlow GPU utilization comparison in 1 GPU(left) and 2 GPU(right)

Conclusion

With increasing model and data sizes, distributed training becomes crucial and each second saved on training can potentially save millions of dollars in cost and reduce carbon emissions effectively. In this work it is shown that PyTorch has better training time (almost half) that that of TensorFlow. It is also evident that data parallelization increases training time by approximately 30% with negligible amount of increase in accuracy. This implies that for the sst2 task of GLUE, BERT produces better performance than that of distributed setting. Hence it is advisable that if training data is not large enough, it is better to train in a single GPU environment and use multi-GPU settings otherwise. More concrete research and analysis can be done on remaining tasks of GLUE so that better statistical evidence can be witnessed for the same.

Future work

Looking towards potential further implementation of our research, we can see a few avenues to pursue. One would be to test on a larger range of GPUs, to collect more data in discovering whether there is more or less difference between and within systems when a larger number of GPUs are used. Another way to extend the comparison would be to include more models to see the interaction with smaller or larger datasets.

Additional work could be conducted to include a wider range of frameworks, such as MXNet or Velox. Unfortunately for the scope of this study, there was not enough time to dedicate to extending the scope to less popular frameworks, but more specific study can be conducted to look at a broader range of libraries, to see the state of progress with multi-GPU and single-GPU efficiency and scalability.

References

[1] Medhat, W., Hassan, A. and Korashy, H., 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, *5*(4), pp.1093-1113.

[2] Freeman, L., 2004. The development of social network analysis. *A Study in the Sociology of Science*, *1*(687), pp.159-167.

[3] Chaudhry, H.N., Javed, Y., Kulsoom, F., Mehmood, Z., Khan, Z.I., Shoaib, U. and Janjua, S.H., 2021. Sentiment analysis of before and after elections: Twitter data of US election 2020. *Electronics*, *10*(17), p.2082.

[4] Xiong, Z., Shen, Q., Xiong, Y., Wang, Y. and Li, W., 2019. New Generation Model of Word Vector Representation Based on CBOW or Skip-Gram. *Computers, Materials & Continua*, *60*(1).

[5] Agarwal, A., Yadav, A. and Vishwakarma, D.K., 2019, May. Multimodal sentiment analysis via RNN variants. In *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)* (pp. 19-23). IEEE.

[6] Li, D. and Qian, J., 2016, October. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)* (pp. 471-475). IEEE.

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.

[8] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[9] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.R., 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

[10] He, P., Gao, J. and Chen, W., 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

[11] Abadi, M., 2016, September. TensorFlow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN international conference on functional programming* (pp. 1-1).

[12] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

[13] Li, H., Kadav, A., Kruus, E. and Ungureanu, C., 2015, April. Malt: distributed data-parallelism for existing ml applications. In *Proceedings of the tenth European conference on computer systems* (pp. 1-16).