Accelerating Large Language Models using CUDA

Koushik Sivarama Krishnan
V01031395
koushik0901@uvic.ca
University of Victoria

ABSTRACT

This report aims to discuss the training methodologies for Large Language Models and the optimization of such processes by training word embeddings and small-scale LLMs with Python and CUDA. By doing so, we seek to familiarize ourselves with the concepts of word embeddings & Large Language Models and how to train them with CUDA and GPUs. We will begin by training the GloVe model[3] using glove-python and NLTK libraries before developing a custom Python model that emulates the GloVe model. We then compare the quality of embeddings from both these methods. We also continued our analysis by hand-coding and training a small-scale LLM using PyTorch based on BERT [1] architecture; we utilized CUDA through PyTorch to improve the training speed. These experiments show the enhancements of the training time and efficiency of these LLMs when CUDA and GPUs are employed. This further supports the reasons for incorporating CUDA in the LLM training processes. In addition to establishing the technical viability of employing CUDA for LLM training, this report has also outlined pertinent and practical difficulties encountered while constructing and training these LLMs.

ACM Reference Format:
Koushik Sivarama Krishnan. 2024. Accelerating Large Language Models using CUDA. In *Proceedings of University of Victoria (Koushik Sivarama Krishnan).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In artificial intelligence, large language models (LLMs) are seen as the new breakthrough through which machines can perceive and produce human language exceptionally. These models are essential for a wide range of processes, from simple tasks such as spell-checking and auto-completion to such ones as translation, content creation, and even programming tasks. However, a large amount of input data has to be processed, and the required computation to train such models is an intricate issue. Therefore, further extension and the corresponding research on improved training methods, especially by utilizing GPUs and the CUDA language, is highly imperative.

The key areas of interest will be the training approaches of word embeddings and small-scale LLMs using Python libraries such as NLTK and PyTorch along with CUDA for computational benefits. The focus is not merely on the acquisition of theoretical knowledge but on the demonstration of how all these technologies can be applied to improve the training of language models.

The word vectors are important parts of the LLMs because they are the first stage that helps the models understand text. Word embeddings focus on converting text data into numbers, something that the model can comprehend. Before the introduction of word embeddings concept, there were other techniques such as Bag of Words and TF-IDF that can convert text into numerical representations. But, compared to these methods, word embeddings captures the context and semantics of the words more accurately and thus improving the overall modelling performance.

As we move towards more complex structures, small scale LLMs are more helpful in the sense that they offer a more easy to handle topographical for conducting experiments and observing the working of larger models. Training these LLMs involves setting up neural network architectures — most commonly transformer — which requires a lot of matrix computations, and in this context, GPUs are effective. This is made possible by PyTorch, which offers a solid and adaptable environment for constructing and training of deep learning models with a first-degree understanding of the processes involved.

Additionally, with the help of Pytorch and Python programming, one can easily optimize the process of moving tensors in and out of GPU's shared memory and performing computations on CUDA cores much more streamlined. Pytorch also abstracts how these tensors are handled at hardware level thus eliminating the need to go back and forth between python and C++ programming. This significantly eliminates bottlenecks and improve performance dramatically.

The utilization of Pytorch, CUDA and Python in the training pipeline of LLMs brings about a dramatic change by offering direct control over operations on the GPU and can reduce training times significantly, making it possible to train models on larger data sets or with more layers within reasonable time frames. This capability is very important, especially in the research and development areas, because training times can have a big impact on performance and invention. Each of these elements are discussed systematically in this report. Beginning from the word embeddings training using NLTK, it then goes on to explain the development and analysis of custom models in Python for similar tasks and finally discusses in detail about the setup and training process of a small-scale LLM using PyTorch.

This report demonstrates the technical possibility of re-adapting previous training processes and structures for LLMs by providing a breakdown of empirical evidence. It offers a comprehensive assessment of the pros and cons and issues that arise during such

Koushik Sivarama Krishnan, July 28 2024, Victoria, BC

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

re-implementations. All the process steps are described and evaluated, which provides a clear picture of the strengths and weaknesses of the existing models.

Thus, this report is not a quest to revolutionize the training of LLMs but rather a quest to better understand the existing models and approaches. By re-implementing these known procedures, we hope to gain a better understanding of LLMs in general how to improve their training times using GPUs.

2 BACKGROUND AND METHODS

The evolution of natural language processing (NLP) technologies has been profoundly influenced by the development of advanced computational models that interpret and generate human language with remarkable precision. Central to these advancements are word embeddings, transformers, and attention mechanisms, each contributing uniquely to the way machines understand text. This section provides a detailed background on these concepts, outlining their development, functionality, and interplay in modern NLP systems.

2.1 Word Embeddings

Advanced deep learning models that can interpret and generate human language with near-perfect accuracy have formed the core of natural language processing. Word embeddings, transformers, and attention mechanisms make a very loud contribution to machines understanding text in their own way. This section provides extended background information on the concepts mentioned above by describing the development, functionality, and interplay in modern NLP systems.

One class of such techniques involves word embeddings, in which words from the vocabulary are mapped into vectors of real numbers; in effect, this achieves a mathematical translation of semantic and syntactic properties of the words the computer is able to understand. This way, it makes possible for similar words to have similar representations in the vector space and brings out the essence of the language in a form that is suitable for algorithmic processing.

The concept of word embeddings was popularized by methods like Word2Vec[2], proposed by Mikolov et al. at Google. A Word2Vec[2] model is a shallow, two-layer neural network trained to reconstruct linguistic contexts of words. Word2Vec[2] may use either one of two architectures: Continuous Bag of Words or Skip-Gram; both have slightly different objectives but are fundamental methods for embedding words in dense vector space based on their context relationships. While CBOW predicts the target word from a given context, Skip-Gram does the reverse: predicting the context given a target word. In such a respect, it will be efficient even on smaller datasets, turning out good on rare words.

Another influential line of word embeddings in the pipeline of improvement is GloVe[3]. This methodology was developed by researchers at Stanford University. While in Word2Vec[2], locally context-driven information is leveraged in the process, GloVe[3] is a count-based model that explicitly constructs a word-context or word-co-occurrence matrix, grounded on statistics across the whole text corpus. After this, matrix factorization methods are used to approximate this matrix and go a level higher in capturing the deeper global relationships between words.

2.2 Transformers

In the transformative paper "Attention is All You Need," Vaswani et al[4]. present the transformer model as an outperforming version of the formerly created sequence processing architectures based on recurrent (RNN) or convolutional (CNN) layers. The transformer avoids recurrent processes by utilizing attention mechanisms which is a more efficient and scalable way to treat sequences.

The architecture of a transformer consists of an encoder and a decoder with layers that mainly work via self-attention mechanisms and feed-forward neural networks. It reads and processes the input sequence through an encoder into a set of attention-driven feature representations that the decoder uses to generate the output sequence in a step-by-step manner.

2.3 Attention Mechanisms

One of the constituent main components of a transformer architecture is the attention [4] mechanism; it helps solve one of the major bottlenecks involved in having to process each word in a sequence one at a time. Attention[4] gives the model the ability to have views or concentrate on different parts of the input sequence as required, regardless of its position in the sequence. This comes in very handy with NLP tasks where the words are relevant to the task and may differ considerably across the sequence.

Specifically, the particular kind of attention used in transformers is what's called "scaled dot-product attention." Basically, it creates three vectors for each input: a query, a key, and a value, often shortcutted as Q, K, and V, respectively. Then, compute an attention score by taking the dot product of the query against all keys, dividing each by their dimensionality's square root to stabilize the learning dynamics, applying a softmax function to obtain weights on the values.

In practice, however, transformers use what is called "multi-head attention," where several attention[4] processes are conducted in parallel. This enables a model to attend to information from different representation subspaces at different positions simultaneously, enhancing its capability of learning from the complex patterns within data.

2.4 BERT

BERT[1] is the latest development in NLP made by Google's researchers in 2018; it stands for Bidirectional Encoder Representations from Transformers. BERT [1]is based on the transformer model but differs in training, which highlights the context in both directions.

In the past, language models processed text data in a one-way fashion either from left to right or from right to left and hence were not very effective in capturing contextual information. BERT [1]solves this problem through a method called the Masked Language Model (MLM). In MLM, some percentage of the input tokens are masked randomly and the model's task is to guess the original identity of the masked words from the context. This makes it possible for BERT[1] to take an in-depth representation of the sentence structure which results to deep understanding of the language.

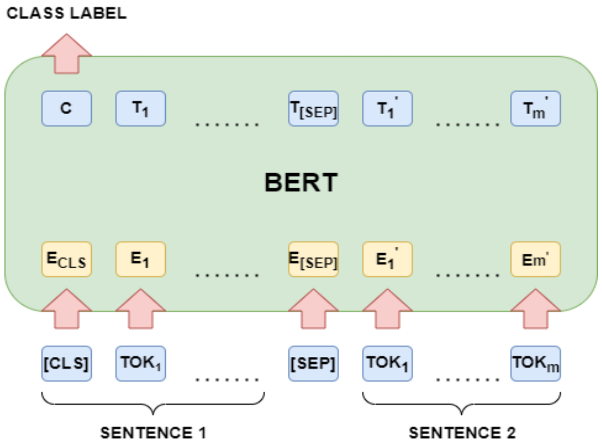

Figure 1: BERT[1] model Architecture

Another new feature of BERT[1] is that during training, it learns so-called “next sentence prediction” (NSP). In NSP, the model learns to predict if a given sentence is relevant to the previous one and hence it is capable of handling tasks that involve the relationships of consecutive sentences such as question answering and natural language inference.

BERT[1] has many layers of transformer blocks that enable text processing in parallel, which is much faster compared to sequential processing models. This architecture makes BERT[1] highly effective for a number of language understanding tasks and it is pre-trained on a massive amount of text thereby making it possible to achieve very good performance on many benchmarks with very minimal task-specific fine tuning.

BERT [1]has revolutionized NLP as it has raised the interest in constructing deeper and bidirectional models. Its success has led to several variants and paved way for more developments in the field and is now recognized as one of the basic models in the current natural language processing technologies.

3 IMPLEMNTATION

3.1 Training GloVe embeddings

In this experiment, we set out to learn about word embeddings and how they work, this we did by creating and training our own GloVe [3]model from basic. Word embeddings are highly beneficial for several NLP applications since they expand words and place similar words in the same vector space. Therefore, using the approach described in this paper, we obtained a better understanding of internal structures of word embeddings and their training.

We initial start with a text corpus for which we performed preprocessing steps to build a vocabulary and word frequency. In this step the text was first converted to all lower case the list was made of the words that made up the text and then the dictionary which contained the words of the text as keys and its index as values. Based on this vocabulary, we constructed the sparse matrix of co-occurrence, which shows the context of words within a certain window size. The formation of this matrix is used as the base for training the GloVe [3]model.

Then, we described the architecture of the GloVe [3]model based on the PyTorch framework. It contains the word and the contextual representations as well as the bias terms for them. These parameters we initialized and for the forward pass we computed the dot product of word and context embeddings adding respective biases. We then created a new loss function for the purpose of comparing the predicted and true co-occurrence counts, weight them to control the frequent and rare co-occurrences.

Due to the large size of the obtained co-occurrence matrix, the process of further analysis required efficient tools to handle sparse matrices and data processing in batches. This approach helped to drastically cut down the memory usage and enabled the model to be trained on a regular computer. The parameters of the model were estimated using the mini-batch update scheme, with the help of which the gradients were calculated by the AdaGrad optimizer. The total training of this model for 4000 epochs took around one minute on Google Colab’s T4 GPU.

After training we extracted the word embeddings and checked the quality of how similar words are embedded using cosine similarity for the closest five words to a given word.

3.2 Training BERT

In this section of the report, the procedure of training a small-scale BERT[1] model from scratch is described to enhance the understanding of the attention mechanism in transformer architecture. This venture offered a wealth of knowledge regarding BERT’s [1]functionality and the overall applicability of attention-based components for describing intricate linguistic phenomena.

The work was started with the creation of the less comprehensive BERT[1] architecture with the fewer transformer blocks and attention heads than in the original model. Based on the PyTorch library with the Python programming language, which is characterized by the ability to effectively work with dynamic graphs, the necessary modules such as custom embedding layers, multi-head attention, and position-wise feed forward layers were developed.

During the training phase, a part of a large text database was used and adapted according to the input format of the model. As mentioned before, during pre-training, both MLM and NSP were used to replicate the two-task learning framework characteristic of BERT[1]. To control the learning rate, the TensorFlow optimiser used a scheduler that incorporated a linear warm up and decay method for improved training. The training of this model took around 10 hours on Google Colab’s T4 GPU.

Validation processes were carried out with specific structured tests to determine the model’s proficiency in token prediction and sentence relationship comprehension. This was accompanied by the use of accuracy and F1 score metrics for the quantitative assessment of performance. Moreover, TensorBoard was used in real-time logging and visualization of both training and validation accuracy.

This particular experiment was crucial in helping understand how exactly PyTorch handles GPU operations, improve the computational throughput, and make model training better. Much attention was paid to the repetitions of the experiments and rational usage of computational resources. The conclusions from this experiment are anticipated to significantly enhance the understanding of the applicability and feasibility of transformer-based models in NLP tasks, not to mention a better understanding of BERT's [1]structure and the operational principles of the attention mechanism.

4 CHALLENGES AND LEARNINGS

One of the major concerns that were observed when training GloVe [3]embeddings was the memory problem that arose when training the co-occurrence matrix especially when dealing with more than fifty thousand words. This matrix was very large and could not be managed and processed within standard computers and as a result the system started to freeze and there were severe problems of performance. To solve this problem, the use of matrixes' sparse representation and the batch processing approaches which reduce memory consumption and enable the training in the general conditions of the computing environment were offered. However, even with these optimizations the model trained on the 50,000 words vocabulary did not yield the results that were presented in the GloVe[3] paper.

This was a limitation because the training process became somewhat tiresome because of the many words that were used in the first list. This adjustment is useful to enable the comparison of the results of the custom implementation with the results of the official GloVe[3] implementation. When training was minimized, the mechanics of the GloVe[3] model were more comprehensible, but the model's functionality of learning a broad spectrum of relation ship semantics was compromised. This process made the need to be selective on the level of complexity of the models to be developed and the amount of resources to be used particularly when developing large scale models in NLP. We also noted that in order to arrive at high levels of accuracy, it is necessary to limit the number of features, that is, the size of the vocabulary in the experiment, while at the same time, monitoring the available computing power.

Fine-tuning a small-scale BERT [1]model had its own problems: One of which was rather remarkable and that was, the model architecture had a lesser number of transformer blocks attention heads than BERT BASE which resulted in the MLM task being insufficient. The model did not learn all the features; therefore, was not capable of performing the MLM task, probably due to the inability of the model to capture some of the more complex structures of a language. Also, it highlighted the problem of large data scarcity for training on Kaggle's GPU, which is also an important point, which indicates the need for large and quality data for BERT model training.

Nevertheless, based on the above analyses, the model had reasonable performance on the NSP task: This gap signified that there was a difference in the requirements of the two tasks as while it is possible to analyse relationships at the sentence level with the aid of a simplified model, at the token level, the model was not enough. This time it was mentioned that although one may make a conclusion that pre-reduction of the BERT model is beneficial for certain applications, one has to embrace that it entails the demise of relatively worse performance on more complex tasks such as MLM.

These experiments also magnified the fact that while training the models there are only two crucial factors that matter that is the architecture of the model and the size of the data set. This training of such complex models such as BERT [1]and GloVe[3] requires a proper balance of the model's size, the computational resources available and the specific task at hand. Reduced models are useful in learning the fundamental concepts of a system; however, the drawback is that the performance of a system on intricate tasks is often poor. A sufficient amount of computational resources and datasets were required to be available, turning into one of the factors that defined the further approach for the experiments.

5 RESULTS

5.1 GloVe Embeddings

The GloVe[3] embeddings were trained on a reduced vocabulary size of 100 words due to computational constraints. The training process involved creating a sparse co-occurrence matrix and using batch processing to handle memory limitations. The primary evaluation metric was cosine similarity between word embeddings, used to identify the top five similar words for a given target word.

The results showed that both the custom and official implementations successfully captured basic semantic relationships. For instance, for the word "court," the closest neighbors were "jury," "superintendent," "commented," and "recommended." The custom GloVe [3]model's performance was comparable to the official implementation, with cosine similarity scores indicating similar embeddings quality. However, the limited vocabulary size and restricted diversity of texts hindered the nuance of the capturing relationships, highlighting the need for large-scale datasets with high diversity for more robust embeddings.

5.2 BERT

The BERT [1]model was trained with a simplified architecture, having 2 transformer blocks and 4 attention heads, on a subset of available datasets. The model's performance was evaluated on two tasks: masked language modeling (MLM) and next sentence prediction (NSP).

For Masked Language Modeling, the model achieved an accuracy of 41% and F1-score of 39%, reflecting difficulty in predicting masked tokens due to limited model complexity and training data. In contrast, the Next Sentence Prediction task yielded an accuracy of 99%, demonstrating better performance in understanding sentence-level relationships. The F1 score for NSP was 0.99%, further indicating satisfactory results in this task.

Monitoring the loss curves of both the Masked Language Modeling and Next Sentence Prediction clearly indicated that despite the small size, it was trying to maximize the learning from the dataset. The comparison of MLM and NSP results underscored the differing demands of these tasks, with the simplified model better suited for sentence-level tasks but inadequate for token-level predictions.

Overall, the experiments with GloVe[3] and BERT [1]provided valuable insights into the mechanics and training requirements of word embeddings and transformer-based models. The results emphasized the critical role of dataset size and model complexity in achieving robust performance in NLP tasks, guiding future efforts to develop more sophisticated and capable NLP models.

6 CONCLUSION

The experiments on GloVe[3] embedding and the BERT model gave a good understanding of the working of word embeddings and the

Experiment	Metric	Result
GloVe	Vocabulary Size	100
	Cosine Similarity - Jury	0.72
	Cosine Similarity - superintendent	0.69
	Cosine Similarity - commented	0.60
	Cosine Similarity - recommended	0.59
BERT Model	MLM Accuracy	41%
	MLM F1 Score	39%
	NSP Accuracy	99%
	NSP F1 Score	99%

Table 1: Results of GloVe and BERT Experiments

transformer architecture. During the training of the GloVe model, it was evident that the problem of having a large vocabulary was a hard problem as well as the need for computational speed when working with large datasets. That is, although the model did not learn many more extensive semantic relationships between the words due to the smaller size of the vocabulary, but this allowed obtaining deeper insights into the mechanisms of learning GloVe [3]embeddings. The BERT model experiment was a good example of how intricate architecture and sufficiently large training data are the keys to high performance, especially when considering such operations as masked language modeling. Thus, the current simplified model showed rather decent performance in the task of next sentence prediction and revealed the fact that various NLP tasks require different levels of complexity. These experiments proved how model complexity data scale, and computational resources are important when developing NLP models. The findings will serve as a basis for improving the further work of increasing the model's complexity and effectiveness, and will extend knowledge in the sphere of natural language processing.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL] https://arxiv.org/abs/1301.3781

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762