

# Benchmarking the Performance of Large Language Models on the Cerebras Wafer Scale Engine

Zuoning Zhang\*, Dhruv Parikh\*, Youning Zhang†, Viktor Prasanna\*

\*University of Southern California

\*{zuoningz, dhruvash, prasanna}@usc.edu †{youningzhang}@berkeley.edu

**Abstract**—Transformer based Large Language Models (LLMs) have recently reached state of the art performance in Natural Language Processing (NLP) and Computer Vision (CV) domains. LLMs use the Multi-Headed Self-Attention (MHSA) mechanism to capture long-range global attention relationships among input words or image patches, drastically improving its performance over prior deep learning approaches.

In this paper, we evaluate the performance of LLMs on the Cerebras Wafer Scale Engine (WSE). Cerebras WSE is a high performance computing system with 2.6 trillion transistors, 850,000 cores and 40 GB on-chip memory. Cerebras WSE’s Sparse Linear Algebra Compute (SLAC) cores eliminate multiply-by-zero operations and its 40 GB of on-chip memory is uniformly distributed among SLAC cores, enabling fast local access to model parameters. Moreover, Cerebras software configures routing between cores at runtime, optimizing communication overhead among cores. As LLMs are becoming more ubiquitous used, new hardware architectures are required to accelerate LLM training and inference. We benchmark the effectiveness of this hardware architecture at accelerating LLM training and inference. Additionally, we analyze if Cerebras WSE can scale the memory-wall associated with traditionally memory-bound compute tasks using its 20 PB/s high bandwidth memory. Furthermore, we examine the performance scalability of Cerebras WSE through a roofline model. By plotting training throughput against computational intensity, we aim to assess their effectiveness at handling high compute-intensive LLM training and inference tasks.

**Index Terms**—large language models, transformers, wafer-scale engine, benchmarking, high performance computing

## I. INTRODUCTION

Large Language Models (LLMs) have been gaining growing interest and led to significant breakthroughs in natural language processing, enabling applications from automated text generation [1] and machine translation [2] to conversational AI [3]. Their ability to understand human inquiry and generate human-like text has enabled advancements in fields such as education [4] and healthcare [5], fostering human and AI to have closer and more direct interactions.

Before transformer models, Recurrent Neural Networks (RNNs) [6] and Long Short-Term Memory (LSTMs) were the state-of-the-art models for sequential data processing tasks. RNNs operate on sequential data processing tasks by maintaining a hidden state that captures information from previous inputs and handles one input element at a time. This enables RNNs to be able to handle inputs of arbitrary length, therefore making it suitable for language-related tasks. RNNs have been deployed in applications such as machine

translation [7] and speech recognition [8]. However, RNNs suffer from vanishing/exploding gradients [9] and struggle at handling inputs that exhibit long-range dependencies among input words. LSTMs address these issues by introducing a gating mechanism, allowing important information to be preserved over long sequences. This makes LSTMs to be more effective at handling input sequences with long-term dependencies [10]. Despite the success of RNNs and LSTMs at handling textual and sequential data, they struggle with parallelization. This limits RNNs and LSTMs from scaling to large model sizes without sacrificing efficiency, therefore making them struggle to handle complex inputs and inputs with extremely long-range relationships. These limitations led to the development of transformer models [1], which utilize a Multi-Head Self-Attention (MHSA) mechanism to process input tokens simultaneously in order to find relationships among input tokens. The parallel nature of MHSA enables the model to efficiently capture relationships among very long input sequences. For example, in GPT-3 [11], the context window size is 2048 tokens, meaning the model will find relationships among the previous 2048 tokens. Moreover, the parallel nature of transformer models allows them to scale in size and excel at handling complex input sentences.

Two cutting-edge transformer models that utilize the MHSA mechanism are BERT [12] and GPT [11] models. Unlike traditional models that process input tokens sequentially (left-to-right or right-to-left), BERT uses transformer encoders, which utilize a bidirectional approach to process input sequences, meaning BERT can calculate MHSA values of a token based on tokens both to the left and right. This enables BERT to have greater contextual understanding of each word and discover more relationships within the input sequence. GPT models utilize transformer decoder components to process texts in a left-to-right manner, where each token can only compute MHSA based on previous tokens in the sequence. This unidirectional nature allows GPT models to predict the next word in a sequence based on the context of all preceding words, making it suitable for autoregressive text generation tasks.

However, the vast scale of these transformer-based LLMs architectures requires significant training and inference costs, presenting challenges in computational and economic resources. In 2018, BERT had a model size of 110M parameters [12]. In 2020, GPT had 175B parameters [11]. The model sizes increased by more than 1000 times in two years. The

growing demand for training larger LLMs requires growing compute intensities and hardware resources. To address these challenges, several new and innovative hardware architectures have been proposed, aiming to optimize the efficiency of LLMs training and inference [13] [14].

One hardware architecture that aims to accelerate the LLMs training and inference tasks is the Cerebras CS-2 system. The Cerebras CS-2 system integrates an entire wafer-scale chip, Cerebras WSE-2, to build a powerful AI accelerator. This enables Cerebras WSE-2 to possess extremely high memory bandwidth and compute intensity. The WSE-2 chip is more than 46000  $mm^2$  in size with 2.6 trillion transistors. The Cerebras WSE-2 architecture is composed of 850,000 cores arranged in a 2-D mesh topology, enabling 20PB/s memory bandwidth and 220Pb/s fabric bandwidth [15] [16].

Analyzing the training throughput and inference latency of LLMs in the Cerebras WSE becomes a critical task as this architecture, with extremely high memory bandwidth, has the potential to allow LLMs training and inference to take advantage of its abundant compute resources without being bottlenecked by memory bandwidth. In this work, we make the following contributions:

- **Training analysis:** We performed in-depth analysis of the training throughput of BERT, on classification task, and GPT-3, on autoregressive text generation task, across different model sizes and batch sizes on the Cerebras WSE platform.
- **Inference analysis:** We perform the end-to-end inference latency analysis for the BERT model on binary classification task on the Cerebras WSE platform across different BERT model sizes and commonly used batch sizes.
- **Projected training epoch duration:** Based on our observed training throughputs, we performed analysis to obtain the projected per-epoch training time to train GPT-3 models and BERT models on the PILE dataset [17] and SST-2 dataset [18].
- **Roofline model:** We completed roofline model analysis for training throughput of BERT and GPT-3 models across different model sizes to gain insight into the scalability of the Cerebras WSE on LLMs training.

Our aim is to deepen our understanding of Cerebras WSE platform’s potential to perform intensive LLMs training and inference tasks.

Section II briefly introduces LLMs and their applications in different fields. Furthermore, this section briefly introduces current state-of-the-art hardware architecture, Cerebras WSE, for LLMs training and inference. Section III describes the models, model sizes, datasets used, and batch sizes for models being evaluated for the experiments, along with the computing platforms on which the experiments were performed. Results are analyzed in section IV. Discussion and conclusion follow in section V.

## II. BACKGROUND

### A. Large Language Models (LLMs)

Large language models have gained growing interest over the past few years. Especially with the release of ChatGPT [19], users began recognizing the capabilities of LLMs and leverage them to assist with daily tasks that were previously time-consuming. Because of its remarkable language understanding abilities, more than one million users signed up to use ChatGPT within one week of its release [20]. Moreover, LLMs have created a profound influence in many industries and fields. For example, LLMs ability to generate and debug code can accelerate the software development process and enhance programmers’ productivity [21]. LLMs can also assist with the creative writing process by offering new ideas and providing feedback and grammatical corrections on input writings of users [20].

Current LLMs use transformer architecture and utilize the Self-Attention (SA) mechanism to efficiently process and understand relationships among words in a sentence. This mechanism enables LLMs to have a powerful understanding of the meaning and context of a given sentence. Mathematically, SA can be defined as the following sequences of operations

$$Q = W^Q X \quad (1)$$

$$K = W^K X \quad (2)$$

$$V = W^V X \quad (3)$$

$$A = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (4)$$

$W^Q \in \mathbb{R}^{d \times d_k}$ ,  $W^K \in \mathbb{R}^{d \times d_k}$ ,  $W^V \in \mathbb{R}^{d \times d_v}$  are the weight matrices for query, key, and value, respectively.  $Q \in \mathbb{R}^{n \times d_k}$ ,  $K \in \mathbb{R}^{n \times d_k}$ ,  $V \in \mathbb{R}^{n \times d_v}$  are the query, key, and value matrices, respectively.  $X \in \mathbb{R}^{n \times d}$  is the input sentence embedding matrix, where  $n$  is the input sequence length,  $d$  is the embedding length of each word in the input sentence. The SA output  $A \in \mathbb{R}^{n \times d_v}$  will then be projected to the same shape as input  $X$  through a feed-forward network. This sequence of operations define the essential elements of a transformer block and is also called a single attention head. In transformer-based LLMs, each transformer block can contain multiple such attention heads. Each head captures different relationships among the tokens in the input sequence, baking within the model a richer context and understanding via the Multi-Headed Self-Attention (MHSA) mechanism. As the input and output dimensions of a transformer block are the same, multiple transformer blocks can be stacked, allowing earlier transformer blocks to provide more context for later blocks.

Transformer models are pre-trained over a large corpus of tokens via tasks such as next word prediction (GPT) [22] and masked language modeling (BERT) [23].

## B. GPT

GPT is a transformer decoder-only architecture where each input only computes SA among previous input tokens. Namely, when computing output SA token  $\mathbf{a}_i \in \mathbb{R}^{d_v}$ , SA only computes among  $\mathbf{q}_i \in \mathbb{R}^{d_k}$  with  $\mathbf{k}_j \in \mathbb{R}^{d_k}$ ,  $\mathbf{v}_j \in \mathbb{R}^{d_v}$  where  $j \leq i$ . This ensures that the model only attends to past and current tokens, not future unseen ones. At each time stamp, GPT computes a probability distribution over the entire vocabulary to predict the probability of next word, enabling it to generate the next word. This process is also called auto-regressive decoding or text generation. Auto-regressive text generation during inference terminates when an *endoftext* token is generated or a predefined maximum output sequence length is reached.

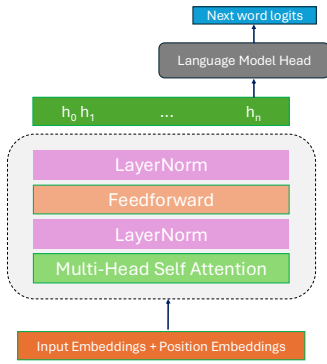


Fig. 1: GPT transformer decoder architecture. Input embeddings are first added with positional embeddings and then fed into GPT transformer-decoder block, where MHSA is performed.

## C. BERT

BERT [12] is a transformer encoder-only architecture where each input computes SA among both previous and future input tokens, called bidirectional SA. Namely, when computing output SA token  $\mathbf{a}_i \in \mathbb{R}^{d_v}$ , SA only computes among  $\mathbf{q}_i \in \mathbb{R}^{d_k}$  with  $\mathbf{k}_j \in \mathbb{R}^{d_k}$ ,  $\mathbf{v}_j \in \mathbb{R}^{d_v}$  where  $j \leq n$ , where  $n$  is the input sequence length. Because SA is computed over the entire input sequence, BERT models have a global comprehensive context over all tokens in the input sequence. Normally, a *[class]* token is added at the beginning of the input sequence. At the output layer, the final hidden state of the *[class]* token is used for the classification of the entire sentence. This ensures that the final classification of the sentence is not biased toward any specific word in the input sequence. BERT models are typically pre-trained on tasks like Masked Language Modeling (MLM) [12], where a random subset of the input sequence is masked and the model predicts the masked words. BERT models are also pre-trained via Next Sentence Prediction (NSP), where the model predicts whether the given two sentences should follow each other. After pre-training, BERT models are fine-tuned for task-specific applications, such as sentiment analysis of input sequence [24].

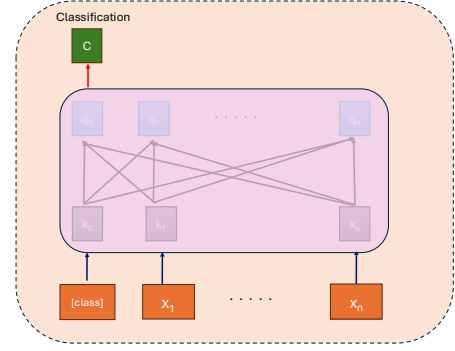


Fig. 2: BERT transformer encoder architecture. *[class]* token embedding is appended to the front of the input sequence. The BERT architecture is similar to the GPT model architecture shown in Figure 1, except that MHSA occurs between the current token and tokens to the left and right. In BERT, the final output hidden state of *[class]* token is used to predict the probability distribution for classification of the input sequence.

## D. Cerebras WSE

All training and inference experiments are conducted on the Cerebras Wafer Scale Engine (WSE) and Cerebras CS-2 system. Cerebras WSE is a powerful AI accelerator built to train LLMs [25]. As of May 2023, Cerebras WSE is the largest chip ever built, at 46,000 mm<sup>2</sup> and containing 2.6 trillion transistors and 850,000 cores. Cerebras WSE's large die size enables greater compute power and enables more computations to occur on-chip [16]. This reduces off-chip communications and enables computation resources to be fully utilized. Cerebras chip is built with Taiwan Semiconductor Manufacturing Company (TSMC) 7 nm process and runs at 1.1 GHz frequency [15]. The WSE's physical design contributes 50% silicon area to static random access memory (SRAM) and 50% to computation logic in each core. Each core features a local 48 KB SRAM that enables the core to have fast access to local memory. In each core, memory is organized into eight 6 KB banks, each 32-bit wide. On top of the local 48 KB memory, each core also has a 256-byte, 64-bit wide, software-configured cache, enabling even faster data access for most frequently used data [15]. This uniformed distributed on-chip memory architecture enables low memory access latency for each core and very high aggregated memory throughput.

Cerebras WSE's fine-grained data flow scheduling, enables cores to only perform computations on non-zero data, saving dynamic power of cores [15]. The combination of this and the high memory bandwidth enables efficient computations on data with unstructured sparsity [26], which is well suited for neural network computations as model weights and input data can possess arbitrary levels of sparsity. Moreover, on top of fine-grained data flow scheduling, each core also has 8 micro-threads holding 8 independent tensor instructions. The scheduler chooses among these cores to execute in the core compute logic, guaranteeing that there is always useful work



#### D. Performance metrics

For training of BERT and GPT-3 models, we use throughput, samples per second, as the performance metric. For inference of BERT models, we use end-to-end latency as the performance metric. Because BERT models perform classification tasks, we define the end-to-end latency of BERT models as the duration from when the input batch is given to the model to when the model output is produced.

### IV. RESULTS

#### A. BERT Training Performance Analysis

Training experiments of BERT models were performed on the BERT-base and BERT-large variants as shown in Table II. Figure 4 shows the training performance, measured in throughput (samples/sec), of the BERT models with varying batch sizes, where the batch size is measured in the number of samples. For both BERT-base and BERT-large, the training throughput increases initially. However, the training throughput of the BERT-base model decreases after reaching the optimal batch size, while the training throughput of the BERT-large model stays roughly the same after a sharp increase in the beginning. The optimal batch size of BERT-base is 2048 and BERT-large is 8192.

In Table IV, we report the projected training time, in seconds, to train one epoch of the SST-2 dataset on the BERT-base and BERT-large models. We calculate the training time based only on the number of training samples, not including test and development sets, and the training throughput we observed in Figure 4.

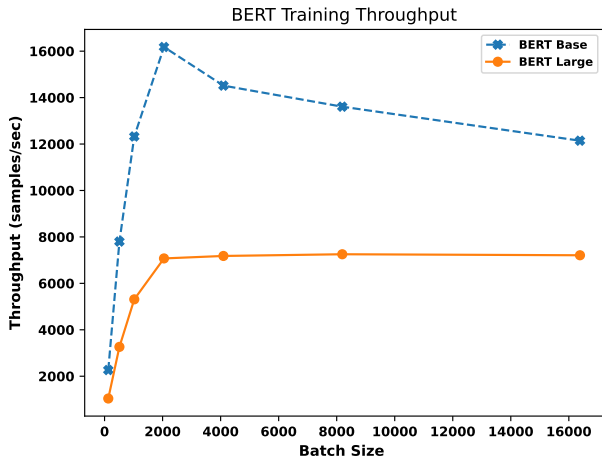


Fig. 4: BERT training throughput analysis. Training throughput is measured in samples/sec. Batch sizes are all powers of 2

#### B. GPT-3 Training Performance Analysis

Training experiments of GPT-3 models were conducted on the GPT-3 model sizes indicated in Table III. Figure 5 shows the training throughput of each GPT-3 model size with varying

TABLE IV: Projected SST-2 One Epoch Training Time (secs)

Batch Size	BERT	
	base	large
128	29.70	64.73
512	8.63	20.64
1024	5.46	12.68
2048	4.16	9.52
4096	4.64	9.38
8192	4.95	9.29
16384	5.55	9.34

batch sizes. For GPT-3 256M model, the training throughput increases as the batch size increases. Larger batch sizes can more effectively utilize the available high bandwidth memory. This facilitates more computations within each core, exploiting the large compute resources available in the WSE. For GPT-3 20B size, training throughput drops at large batch sizes. This is due to the fact that the memory bandwidth on the chip is fully utilized and data transfer dominates the compute time during training. For other GPT-3 model sizes, the training throughput stays roughly the same across model sizes. This highlights the WSE’s unique capacity to scale training for large models and batch sizes without a drop in throughput.

In Table V, we provide the projected training time, in hours, to train one epoch of the entire PILE dataset [27]. The projected training time includes one pass of the train, test, and development sets. We calculate the training time based on the training throughput we observed in Figure 5 and the total number of samples reported in the PILE datasheet, 211,043,181 samples. We provide the training time for batch size of 128 and 256 as the throughput is similar for other batch sizes and 128 and 256 are common choices of batch size.

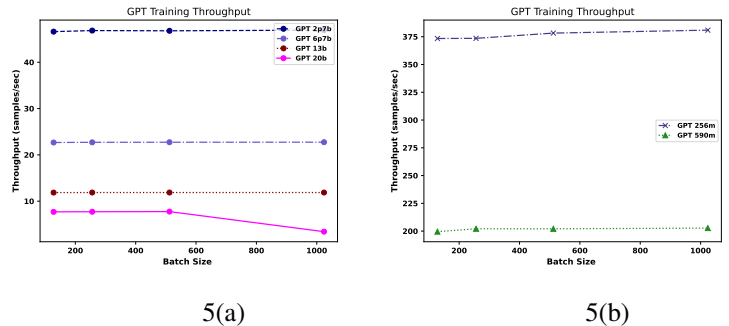


Fig. 5: Training throughput analysis of GPT-3 models over varying batch sizes. 5(a) shows the training throughput for GPT-3 2.7B, 6.7B, 13B, and 20B models. 5(b) shows training throughput for GPT-3 256M and 590M models. All batch sizes are measured in number of samples and are power of 2

TABLE V: Projected PILE One Epoch Training Time (hours)

Model	Batch Size	
	128	256
256 M	157	157
590 M	294	290
2.7 B	1258	1252
6.7 B	2586	2581
13 B	4943	4939
20 B	7613	7594

C. BERT Inference Performance Analysis

Figure 6 shows the inference latency for BERT models over varying batch sizes. Our experiment results indicate that the BERT-base model latency does not vary by much for all batch sizes except batch size of 1. Likewise, BERT-large model latency does not change much after a certain batch size. However, recall that end-to-end latency is measured as the duration it takes for the output of the entire batch to be produced. Therefore, this indicates that performing inference for larger batch sizes is beneficial on the Cerebras WSE platform as the end-to-end latency does not vary by much with increasing batch size. This reduces the average inference latency per sample for larger batch sizes, as is expected for high-compute high-bandwidth systems like Cerebras.

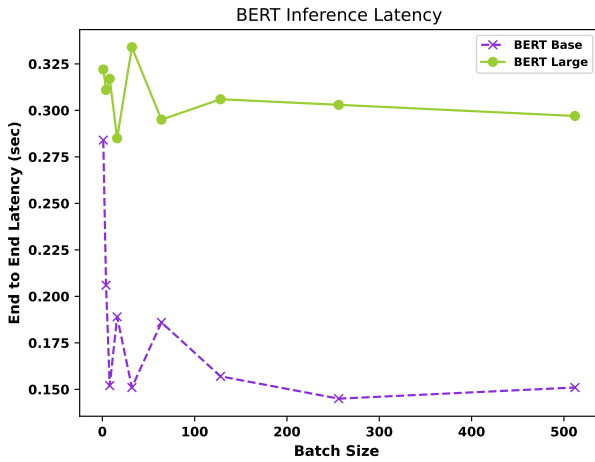


Fig. 6: BERT end-to-end inference latency analysis.

D. Roofline Models

Based on our experimental results, we completed roofline model analysis for BERT and GPT-3 training, shown in Figure 7 and Figure 8. Our results show that both BERT-base and BERT-large training on all the batch sizes we investigated operate in the compute-bound region. Moreover, all GPT-3 training for batch size of 128 operate in the compute-bound region. This highlights that training large models on the Cerebras WSE is scalable, with Cerebras WSE scaling the memory wall for LLM training through its unique architecture.

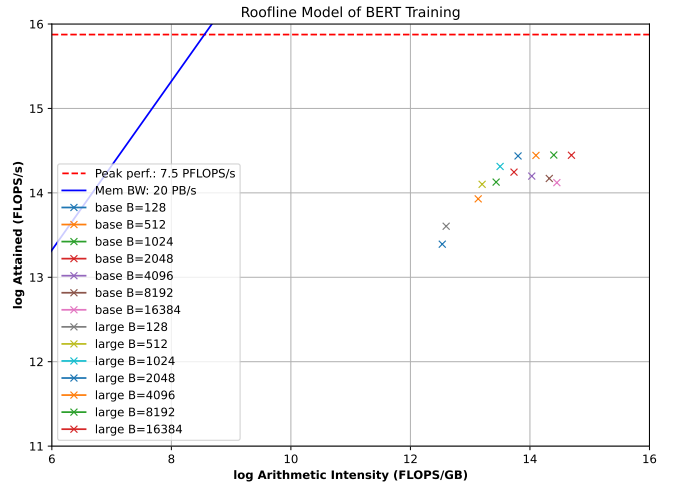


Fig. 7: BERT roofline analysis. Axes are in log scale.

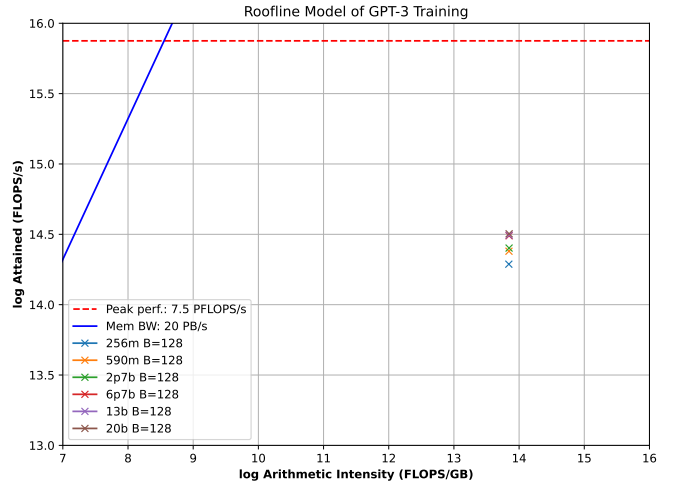


Fig. 8: GPT-3 roofline analysis. Axes are in log scale. Only batch size of 128 is investigated.

V. CONCLUSION AND FUTURE WORK

In this paper, we examined the training throughput of BERT and GPT-3 training across commonly used batch sizes and model sizes, as well as inference latency for BERT. We also investigated the roofline model of BERT and GPT-3 training, gaining insights in the scalability and potential of the Cerebras WSE for accelerating LLM training and inference. Investigating Cerebras WSE’s performance for more models, especially computer vision models, will be a promising area for exploration because of its widespread applications.

ACKNOWLEDGMENT

This work was performed on a CS-2 platform at the University of Southern California supported by the US Army DURIP program. This work was also supported in part by DEVCOM Army Research Lab (ARL) under grant W911NF2320186.

We would like to thank the USC Center for Advanced Research Computing (CARC) for their support in installing and accessing the CS-2 platform and the Cerebras staff for their continued engineering support and helpful discussions.

## REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [2] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," *Advances in neural information processing systems*, vol. 30, 2017.
- [3] Z. Xue, R. Li, and M. Li, "Recent progress in conversational ai," *arXiv preprint arXiv:2204.09719*, 2022.
- [4] Z. Zhang, D. Zhang-Li, J. Yu, L. Gong, J. Zhou, Z. Liu, L. Hou, and J. Li, "Simulating classroom education with llm-empowered agents," *arXiv preprint arXiv:2406.19226*, 2024.
- [5] S. A. Gebreab, K. Salah, R. Jayaraman, M. H. ur Rehman, and S. Ellaham, "Llm-based framework for administrative task automation in healthcare," in *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, 2024, pp. 1–7.
- [6] J. Elman, "Finding structure in time. cognitive science, 14 (2), 179–211." 1990.
- [7] J. R. Chaudhary and A. C. Patel, "Bilingual machine translation using rnn based deep learning," *Int J Sci Res Sci Eng Technol*, vol. 4, no. 4, pp. 1480–1484, 2018.
- [8] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE workshop on automatic speech recognition and understanding (ASRU)*. IEEE, 2015, pp. 167–174.
- [9] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.
- [10] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.
- [11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [13] W. Luk, K. F. C. Yiu, R. Li, K. Mishchenko, S. I. Venieris, H. Fan *et al.*, "Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference," *arXiv preprint arXiv:2405.18628*, 2024.
- [14] Y. Huang, L. J. Wan, H. Ye, M. Jha, J. Wang, Y. Li, X. Zhang, and D. Chen, "New solutions on llm acceleration, optimization, and application," *arXiv preprint arXiv:2406.10903*, 2024.
- [15] S. Lie, "Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning," *IEEE Micro*, vol. 43, no. 3, pp. 18–30, 2023.
- [16] G. Lauterbach, "The path to successful wafer-scale integration: The cerebras story," *IEEE Micro*, vol. 41, no. 6, pp. 52–57, 2021.
- [17] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.
- [18] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.
- [19] T. OpenAI, "Chatgpt: Optimizing language models for dialogue. openai," 2022.
- [20] M. Abdullah, A. Madain, and Y. Jararweh, "Chatgpt: Fundamentals, applications and social impacts," in *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. Ieee, 2022, pp. 1–8.
- [21] Q. Gu, "Llm-based code generation method for golang compiler testing," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 2201–2203.
- [22] M. Geva, A. Caciularu, K. R. Wang, and Y. Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," *arXiv preprint arXiv:2203.14680*, 2022.
- [23] J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff, "Masked language model scoring," *arXiv preprint arXiv:1910.14659*, 2019.
- [24] M. G. Sousa, K. Sakiyama, L. de Souza Rodrigues, P. H. Moraes, E. R. Fernandes, and E. T. Matsubara, "Bert for stock market sentiment analysis," in *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*. IEEE, 2019, pp. 1597–1601.
- [25] N. Sengupta, S. K. Sahu, B. Jia, S. Katipomu, H. Li, F. Koto, O. M. Afzal, S. Kamboj, O. Pandit, R. Pal *et al.*, "Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models," *arXiv preprint arXiv:2308.16149*, 2023.
- [26] V. Thangarasa, M. Salem, S. Saxena, K. Leong, J. Hestness, and S. Lie, "Mediswift: Efficient sparse pre-trained biomedical language models," 2024. [Online]. Available: <https://arxiv.org/abs/2403.00952>
- [27] S. Biderman, K. Bicheno, and L. Gao, "Datashet for the pile," *arXiv preprint arXiv:2201.07311*, 2022.