National University of Computer and Emerging Sciences

Submitted By:

Name:	**Reg. No:**
Menaal Maqbool	23L-8052

Department of Data Science

FAST-NU, Lahore, Pakistan

Application of Various Compression and Acceleration Techniques of Pre-trained Language Models

1. Abstract:

The increasing demand for deploying large pre-trained models like BERT in resource-constrained environments has led to the exploration of various model optimization techniques. This report investigates four such techniques—Low-Rank Adaptation (LoRA), Pruning, Quantization, and Knowledge Distillation—aimed at improving the efficiency of BERT while maintaining competitive performance. The LoRA technique reduced the number of trainable parameters by 99.73%, achieving substantial savings in computational resources with only a slight drop in accuracy. Pruning was applied to the Linear layers of the BERT model, reducing model complexity but leading to a small accuracy decrease. Quantization converted the model from float32 to int8, resulting in a 58% reduction in model size and a notable improvement in inference speed with minimal accuracy loss. Finally, Knowledge Distillation transferred knowledge from a teacher model (DistilBERT) to a smaller student model, maintaining high accuracy while significantly reducing model size. The results demonstrate the feasibility of optimizing BERT for deployment in resource-limited settings, with trade-offs between performance, size, and speed. Future work will focus on refining these techniques and exploring their combination for enhanced model efficiency.

2. Introduction and Problem Statement:

In natural language processing (NLP), models like BERT have shown remarkable success across various tasks. However, they come with significant computational overhead, which can limit their deployment in resource-constrained environments. This report explores four model optimization techniques—**LoRA (Low-Rank Adaptation)**, **Pruning**, **Quantization**, and **Knowledge Distillation**—to make BERT-based models more efficient without sacrificing too much accuracy. The goal is to reduce model size, improve inference speed, and maintain competitive performance across these techniques.

3. Literature Review:

In the ever-evolving field of Natural Language Processing (NLP), transformer models have emerged as a game changer, changing the way machines can understand and generate human language. These models were first introduced in the paper *Attention Is All You Need* by [1] and were specifically designed to work with sequential data, transformers can handle entire sequences simultaneously. This fundamental shift allows them to compute complex dependencies between words in an efficient and parallelized manner. Since then, transformers have been extensively applied across various NLP tasks, including machine translation, text summarization and sentiment analysis [2]. Models like BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), T5 (Text-to-Text Transfer Transformer), have demonstrated state-of-the-art performance on a wide range of NLP benchmarks.

Their ability to capture contextual relationships within text has led to remarkable improvements in NL understanding and generation. Although these models are very efficient but they have high computational costs associated with them, which presents a significant challenge [3]. Large-scale Pre-trained Language Models (PLMs) require substantial hardware resources, making them expensive to train and deploy. These models' increasing complexity and size, with billions or even trillions of parameters, have driven the need for compression and acceleration techniques to make them more efficient and accessible [4]. To address these challenges, researchers have explored various strategies for optimizing PLMs while maintaining their predictive performance. These include model pruning [2], quantization [5], knowledge distillation [5], and low-rank factorization [6].

The techniques that aim to make the ML models faster (time-efficient), consume fewer computational resources (computation-efficient), less memory, and less disk space are referred to as ***efficient inference*** [7]. Model compression and acceleration techniques play a crucial role in achieving efficient inference. For our term project, we aim to apply various model compression and acceleration techniques and conduct a comparative analysis. However, since these techniques are often applied in isolation, consolidating approaches and defining universal

evaluation metrics pose challenges. Several studies have explored methods to optimize PLMs while maintaining their predictive performance.

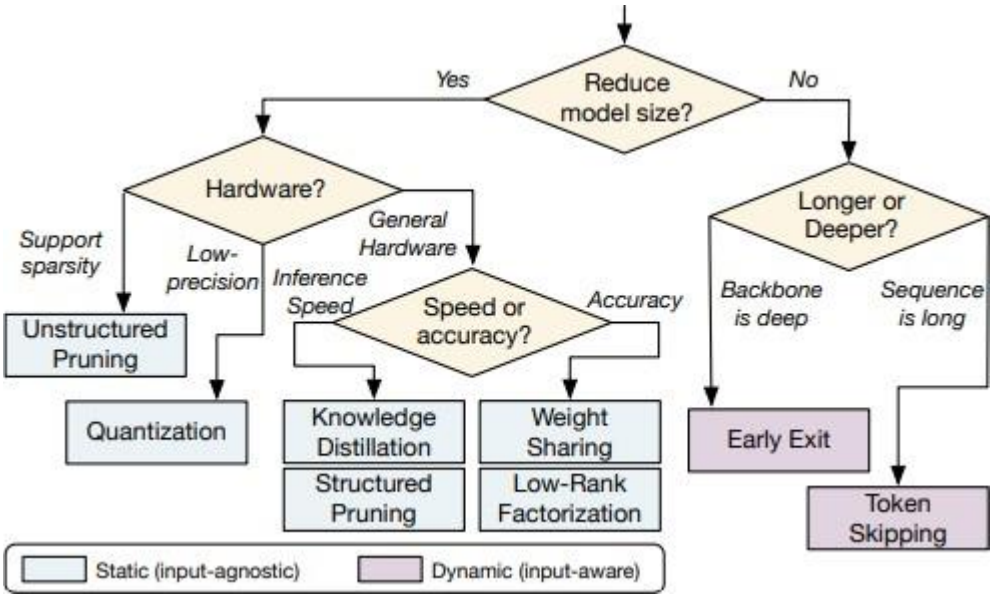

Figure 1 An oversimplified decision tree for PLM model compression and acceleration

Figure 1 presents an oversimplified decision tree for PLM model compression and acceleration. We aim to implement at least one technique from each category and assess their impact on PLMs. Some of the selected techniques include:

3.1. Model pruning:

Model pruning removes unnecessary parameters from a trained model to reduce the size and computational cost while maintaining accuracy [8]. Pruning is a process for enhancing ML model efficiency and effectiveness. By systematically removing less significant parameters of a DNN, pruning reduces the model's size and computational complexity without substantially compromising its performance [9]. This practice is especially vital in contexts where storage and computational resources are limited. Figure 2 depicts an illustration of a pruned DNN.

- **Unstructured Pruning**: Removes individual weights based on their importance, often using magnitude-based pruning. Unstructured pruning removes "unimportant" connections in a neural network by setting the corresponding weights to 0 [10]. After

pruning, the weight matrix often becomes sparse. To exploit the characteristics of a sparse matrix to achieve computation and memory efficiency, specialized hardware (e.g., sparse tensor cores in Nvidia A100 and software (e.g., PyTorch Sparse API2) are necessary.

- **Structured Pruning**: Structured pruning removes weight blocks, rows, attention heads, or layers from a model. Compared to unstructured pruning, it can usually achieve acceleration and memory reduction without specialized hardware or software [11].

Attention head pruning in Transformers reduces computational overhead while keeping model performance close to the original.

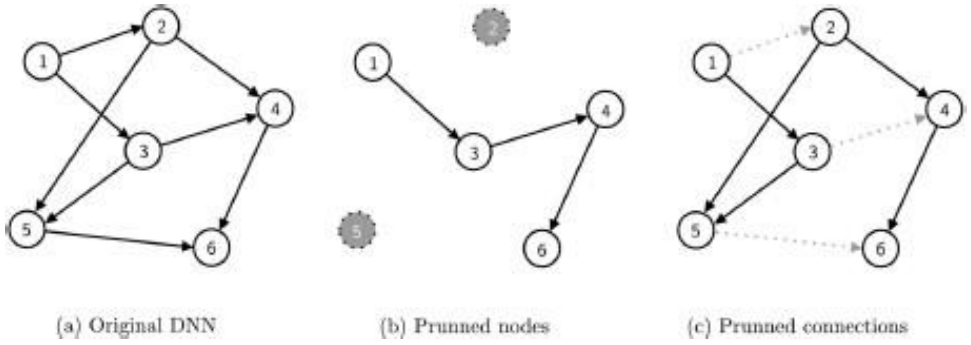

Figure 2 The process of weight pruning in a DNN. (a) Shows the original DNN with all nodes and connections intact. (b) Highlights the nodes that have been pruned, indicating the parts of the network identified as non-essential. (c) Displays the pruned connection [12]

It is commonly used in CNNs and targets neurons that contribute less to the network's ability to model the problem. By removing these neurons, the network's complexity is reduced, potentially leading to faster inference times [12].

3.2. Quantization

Quantization reduces the precision of the numerical values (weights and activations) within a neural network, such as converting 32-bit floating-point numbers to 8-bit integers. This process *decreases the model's memory footprint and accelerates inference* by enabling the use of more efficient hardware operations [13]. Quantization techniques, such as those applied in CoFi [11],

convert high-precision floating-point computations into lower-bit representations, reducing memory consumption and accelerating inference. Quantization aims to reduce the total number of bits required to represent each parameter, usually converting floating-point numbers into integers [14].

3.3. Knowledge distillation

Knowledge distillation aims to transfer the knowledge from a large teacher model to a small student model. It has been widely used to compress large-scale pre-trained language models (PLMs) like BERT [15] and RoBERTa [16] in recent years. Knowledge distillation involves training a smaller, more efficient "student" model to replicate the behavior of a larger, complex "teacher" model. The student model learns to mimic the teacher's outputs, capturing its knowledge in a compressed form [17]. The teacher model generates output probabilities or logits for a given input. The student model is trained to match these outputs, often using a combination of the original training loss and a distillation loss that measures the difference between the teacher's and student's outputs. DistilBERT is a distilled version of the BERT model, achieving approximately 60% of BERT's size while retaining about 97% of its language understanding capabilities [6].

3.4. Low rank factorization:

Low-rank factorization is a way to make NN smaller and simpler without making it less effective. This method focuses on decomposing large, dense-weight matrices found in DNN into two smaller, lower-rank matrices. The essence of low-rank factorization is its ability to combine these two resulting data matrices to approximate the original, thereby achieving compression. This method reduces the model size and data processing demands, making CNN more suitable for applications in resource-limited environments [18]. In the context of Convolutional Neural Networks (CNNs) and Transformer-based architectures, low-rank factorization has been successfully applied to compress layers without sacrificing accuracy. Methods such as Singular Value Decomposition (SVD), Tucker decomposition, and tensor factorization have been explored to optimize deep models while maintaining their expressive power [19].

3.5. Token Skipping:

Token skipping is an effective strategy for reducing the computational cost of transformer-based models while maintaining competitive accuracy. These methods dynamically eliminate redundant or less informative tokens during inference, thereby enhancing efficiency. PoWER-BERT (Progressive Word-vector Elimination for BERT) was proposed by [20]. PoWER-BERT introduces a mechanism to progressively remove tokens deemed less important, significantly improving inference time without substantial loss in model performance. This method is particularly valuable for real-time NLP applications, where rapid response times are crucial.

Recent surveys provide comprehensive overviews of various model compression and acceleration strategies [21]. These studies highlight the trade-offs between efficiency and accuracy, offering insights into how token pruning can optimize transformer-based architectures. By systematically comparing different techniques, these works form the basis for evaluating the effectiveness of token skipping relative to other compression methods, such as quantization, pruning, and distillation.

Pre-training of the BERT model will be achieved on the original datasets on which BERT was trained, specifically the Wikipedia dumps and BookCorpus. These large-scale textual resources provide extensive linguistic diversity and contextual richness, which are crucial for learning high-quality language representations. For fine-tuning, the model will be adapted to various downstream NLP tasks, including:

- **Text Classification** – used for sentiment analysis, spam detection, and topic categorization.
- **Named Entity Recognition (NER)** – enables the identification of key entities in a text such as names, locations, and dates.
- **Question Answering (Q/A)** – utilized in conversational AI and search engines to provide precise answers.
- **Text Summarization** – helps generate concise versions of longer texts while maintaining essential information.

To assess the efficiency of compression and acceleration techniques that were discussed above, various metrics are considered:

3.6. Floating Point Operations (FLOPs):

This is a key metric used to measure the computational cost of running a model. It counts the number of floating-point arithmetic operations (such as addition, multiplication, and division) needed for inference. To assess the efficiency of compression and acceleration techniques in pre-trained language models (PLMs), FLOPs serve as a key metric for computational cost. High FLOPs lead to slower inference and higher energy consumption, making deployment challenging. Techniques like quantization lower numerical precision, pruning removes redundant parameters, and knowledge distillation transfers knowledge to a smaller model. These optimizations enhance inference efficiency, making PLMs more viable for real-world applications.

3.7. Inference Time:

Latency measures the time a model takes to generate predictions, directly impacting real-time applications. High latency in pre-trained language models like BERT arises from complex computations, including self-attention and large feedforward layers. Techniques such as pruning, quantization, and knowledge distillation help reduce latency by streamlining computations. Reducing latency is crucial for deploying PLMs in real-time systems, including chatbots, voice assistants, and mobile applications.

3.8. Speed-up Ratio:

Speed-up ratio quantifies the improvement in inference speed by comparing a compressed or accelerated model to a baseline model. A higher speed-up ratio indicates a more optimized model with reduced computational overhead, making deployment on resource-limited devices more feasible.

3.9. Number of Parameters & Model Size:

The number of parameters and overall model size determine memory usage and computational feasibility. Large models like BERT require significant storage and processing power, limiting

their deployment on edge devices. Smaller model sizes enable faster inference, lower memory consumption, and efficient deployment in real-world applications.

Performance evaluation of compressed and accelerated pre-trained language models is crucial to ensure that reductions in model size and computational requirements do not compromise their effectiveness. Standardized benchmarks have been developed to assess various aspects of these models, focusing on both their linguistic capabilities and operational efficiency.

3.10. General Language Understanding Evaluation (GLUE):

GLUE is a comprehensive benchmark designed to evaluate the generalization and robustness of natural language understanding models across a diverse set of tasks [22]. It comprises nine distinct tasks, including:

- Corpus of Linguistic Acceptability (CoLA): Assesses the grammaticality of sentences.
- Stanford Sentiment Treebank (SST-2): Evaluates sentiment analysis.
- Microsoft Research Paraphrase Corpus (MRPC): Tests paraphrase detection.
- Semantic Textual Similarity Benchmark (STS-B): Measures the similarity between sentence pairs.
- Quora Question Pairs (QQP): Identifies duplicate questions.
- Multi-Genre Natural Language Inference (MNLI): Evaluates understanding across genres.
- Question Natural Language Inference (QNLI): Focuses on question-answer entailment.
- Recognizing Textual Entailment (RTE): Determines textual entailment.
- Winograd Schema Challenge (WNLI): Tests coreference resolution.

By providing a unified evaluation platform, GLUE encourages the development of models that can perform well across multiple tasks, promoting the creation of general-purpose language understanding systems [23].

3.11. EfficientQA:

EfficientQA is a benchmark and competition that focuses on open-domain question-answering systems, emphasizing the balance between accuracy and efficiency. Participants are challenged to develop models capable of providing precise answers to questions while adhering to strict computational constraints, such as limited memory usage and processing time [24]. This benchmark is particularly relevant for evaluating compressed and accelerated PLMs, it highlights the trade-offs between model performance and resource consumption.

3.12. SuperGLUE:

Building upon the original GLUE benchmark, SuperGLUE offers a more challenging suite of tasks designed to push the boundaries of natural language understanding models. It includes tasks that require advanced reasoning, common sense knowledge, and understanding of nuanced language structures [25]. SuperGLUE serves as a rigorous testbed for evaluating the performance of compressed and accelerated PLMs on complex language tasks.

These benchmarks collectively provide a comprehensive framework for evaluating the performance, efficiency, and robustness of compressed and accelerated PLMs, guiding the development of models that are both effective and resource-efficient [26].

The rapid advancements in transformer-based models have significantly improved NLP applications, but their increasing computational costs and memory requirements present major challenges for deployment, particularly in resource-constrained environments. To address this issue, various compression and acceleration techniques, including model pruning, quantization, knowledge distillation, low-rank factorization, and token skipping, have been explored to enhance model efficiency while maintaining predictive performance. Despite substantial progress, selecting the optimal combination of these techniques remains an open research problem. Additionally, evaluating their effectiveness requires standardized metrics such as FLOPs, inference time, speed-up ratio, and model size, along with benchmarking frameworks like GLUE, SuperGLUE, and EfficientQA.

4. Methodology:

Each technique employed aims at reducing the computational burden while keeping the model's effectiveness intact. Below is an overview of the techniques tested:

4.1. LoRA (Low-Rank Adaptation on BERT):

- **LoRA** was used to reduce the number of trainable parameters in BERT, resulting in significant computational and memory savings while maintaining performance.
- **Fine-tuned Model:** Only 0.27% of the parameters were updated, with the rest of the model frozen.
- **Task:** Sentiment classification on the **SST-2** dataset from GLUE.

4.2. Pruning:

- **Global Unstructured Pruning** was applied to the BERT-base model using the **L1Unstructured** method.
- **Pruning Target:** 20% of the weights from the Linear (fully connected) layers were pruned.
- **Result:** Pruning led to a slight degradation in accuracy, but reduced model complexity.

4.3. Quantization:

- **Dynamic Quantization** was performed on BERT's Linear layers by converting the model weights from **float32 to int8**.
- **Result:** Model size was reduced by approximately 58%, and inference time improved without a major loss in accuracy.

4.4. Knowledge Distillation:

- A smaller **student model** (DistilBERT) was trained to mimic the behavior of a pre-trained **teacher model** (DistilBERT).

- **Loss Function:** A combination of **hard loss** (cross-entropy) and **soft loss** (KL divergence between teacher and student logits) was used to train the student.
- **Dataset:** GLUE SST-2 for sentiment analysis.

5. Implementation Details:

5.1. LoRA (Low-Rank Adaptation on BERT)

- **Base Model:** Pre-trained **BERT** model fine-tuned on SST-2.
- **LoRA Model:** Fine-tuned only 0.27% of the model's parameters.
- **Result:** The LoRA model achieved **89.10% accuracy** (slightly lower than the base model's 91.51%), but with major savings in model size and computation.

5.2. Pruning Experiment

- **Model:** BERT-base model fine-tuned for a single batch of SST-2.
- **Pruning Method: L1Unstructured** pruning, targeting **20% of Linear layer weights**.
- **Effect:**
  - Loss increased slightly, and accuracy dropped by about 3% from 68.74% to 65.74%.
  - This result suggests pruning had a mild negative impact on performance but reduced model complexity.

5.3. Quantization Experiment

- **Model:** BERT-base model with dynamic quantization on its Linear layers.
- **Process:** Conversion from **float32 to int8**.
- **Result:**
  - Model size reduced by **58%** (from 438MB to 181MB).
  - Inference time improved from **158 seconds to 114 seconds**.

- The impact on accuracy was minimal, with the model maintaining comparable performance after quantization.

5.4. Knowledge Distillation Experiment

- **Teacher Model:** Pre-trained **DistilBERT**.
- **Student Model:** A smaller **DistilBERT** model trained using knowledge distillation.
- **Training Process:** Soft labels from the teacher model were combined with hard labels (true labels from SST-2) to train the student model.
- **Result:** The student model achieved a validation accuracy of **0.8976** (approximately 90%) on the SST-2 dataset, with a slight loss compared to the teacher's performance.

6. Results and Discussion:

6.1. LoRA (Low-Rank Adaptation on BERT)

- **Validation Accuracy:** The LoRA model achieved an accuracy of **89.10%**, which was a slight drop from the **91.51%** accuracy of the base model.
- **Key Advantage:** The LoRA model trained only **0.27%** of the parameters, resulting in substantial computational savings while retaining most of the performance.

6.2. Pruning

- **Accuracy:** The pruned model showed a **3% drop in accuracy**, from **68.74%** to **65.74%**.
- **Key Insight:** While pruning reduced model complexity by removing 20% of the Linear layer weights, it resulted in a noticeable drop in accuracy. This experiment highlights the trade-off between model size reduction and accuracy loss.

6.3. Quantization

- **Model Size Reduction:** The model size was reduced by **58%**, significantly improving memory and storage efficiency.

- **Inference Speed:** Inference time improved from **158 seconds** to **114 seconds**, making the quantized model more suitable for real-time applications.
- **Accuracy:** The accuracy was largely maintained, with only a minor impact, demonstrating that quantization can effectively reduce model size and inference time without significantly affecting performance.

6.4. Knowledge Distillation

- **Validation Accuracy:** The student model achieved **0.8976** validation accuracy, just slightly behind the teacher model. The loss in performance was minimal, with the benefit of a smaller, faster model.
- **Key Insight:** Knowledge distillation helped compress the model effectively, reducing size and complexity while maintaining high performance. The student model can be deployed in environments where resource efficiency is crucial.

7. Conclusion and Future Work:

7.1. Conclusion

- **LoRA:** Offers a promising approach for reducing the number of trainable parameters, achieving large compute savings with a minor drop in performance.
- **Pruning:** While pruning provides model size reduction, it can cause a significant drop in performance, especially when the pruning rate is high.
- **Quantization:** Effectively reduces model size and inference time, with minimal impact on accuracy, making it ideal for resource-constrained environments.
- **Knowledge Distillation:** Successfully compresses the model while maintaining high accuracy. The distilled model can be deployed in production environments with resource constraints.

7.2. Future Work

- **LoRA:** Experiment with different configurations of LoRA to find an optimal trade-off between model size and performance.

- **Pruning:** Investigate other pruning strategies and apply pruning gradually across more layers to mitigate accuracy loss.
- **Quantization:** Test quantization on a larger scale and explore its application in distributed environments.
- **Distillation:** Train the student model for longer periods to achieve performance closer to the teacher model and experiment with other distillation methods.

References:

[1] A. Vaswani *et al.*, "Attention Is All You Need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, pp. 5999–6009, Jun. 2017, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/1706.03762v7.

[2] C. Xu and J. McAuley, "A Survey on Model Compression and Acceleration for Pretrained Language Models," *Proc. 37th AAAI Conf. Artif. Intell. AAAI 2023*, vol. 37, pp. 10566–10575, Feb. 2022, doi: 10.1609/aaai.v37i9.26255.

[3] Y. J. Chang, D. Y. Hong, P. Liu, and J. J. Wu, "Efficient Inference on Convolutional Neural Networks by Image Difficulty Prediction," *Proc. - 2022 IEEE Int. Conf. Big Data, Big Data 2022*, pp. 5672–5681, 2022, doi: 10.1109/BIGDATA55660.2022.10020754.

[4] I. Turc, M.-W. Chang, K. Lee, K. Toutanova, and G. Research, "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models," Aug. 2019, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/1908.08962v2.

[5] M. Xia, Z. Zhong, and D. Chen, "Structured Pruning Learns Compact and Accurate Models," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, vol. 1, pp. 1513–1528, Apr. 2022, doi: 10.18653/v1/2022.acl-long.107.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," Oct. 2019, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/1910.01108v4.

[7] Z. Zhou *et al.*, “A Survey on Efficient Inference for Large Language Models,” Apr. 2024, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/2404.14294v3.

[8] U. Bibi *et al.*, “Advances in Pruning and Quantization for Natural Language Processing,” *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3465631.

[9] A. See, M. T. Luong, and C. D. Manning, “Compression of Neural Machine Translation Models via Pruning,” *CoNLL 2016 - 20th SIGNLL Conf. Comput. Nat. Lang. Learn. Proc.*, pp. 291–301, Jun. 2016, doi: 10.18653/v1/k16-1029.

[10] A. Mishra *et al.*, “Accelerating Sparse Deep Neural Networks,” *arXiv*, Apr. 2021, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/2104.08378v1.

[11] Z. Wang, J. Wohlwend, and T. Lei, “Structured Pruning of Large Language Models,” *EMNLP 2020 - 2020 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 6151–6162, Oct. 2019, doi: 10.18653/v1/2020.emnlp-main.496.

[12] P. V. Dantas, W. Sabino da Silva, L. C. Cordeiro, and C. B. Carvalho, “A comprehensive review of model compression techniques in machine learning,” *Appl. Intell. 2024 5422*, vol. 54, no. 22, pp. 11804–11844, Sep. 2024, doi: 10.1007/S10489-024-05747-W.

[13] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, “SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models.” PMLR, pp. 38087–38099, Jul. 03, 2023, Accessed: Feb. 20, 2025. [Online]. Available: https://proceedings.mlr.press/v202/xiao23c.html.

[14] J. Kim, S. Chang, and N. Kwak, “PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation,” *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 3, pp. 1863–1867, Jun. 2021, doi: 10.21437/Interspeech.2021-248.

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[16] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” Jul. 2019, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/1907.11692v1.

[17] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021, doi: 10.1007/S11263-021-01453-Z/METRICS.

[18] G. Cai, J. Li, X. Liu, Z. Chen, and H. Zhang, “Learning and Compressing: Low-Rank Matrix Factorization for Deep Neural Network Compression,” *Appl. Sci. 2023, Vol. 13, Page 2704*, vol. 13, no. 4, p. 2704, Feb. 2023, doi: 10.3390/APP13042704.

[19] Y. C. Hsu, T. Hua, S. E. Chang, Q. Lou, Y. Shen, and H. Jin, “Language model compression with weighted low-rank factorization,” *ICLR 2022 - 10th Int. Conf. Learn. Represent.*, Jun. 2022, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/2207.00112v1.

[20] S. Goyal, A. R. Choudhury, S. M. Raje, V. T. Chakaravarthy, Y. Sabharwal, and A. Verma, “PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination,” *37th Int. Conf. Mach. Learn. ICML 2020*, vol. PartF168147-5, pp. 3648–3657, Jan. 2020, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/2001.08950v5.

[21] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient Transformers: A Survey,” *ACM Comput. Surv.*, vol. 55, no. 6, Sep. 2020, doi: 10.1145/3530811.

[22] L. Yang *et al.*, “GLUE-X: Evaluating Natural Language Understanding Models from an Out-of-distribution Generalization Perspective,” *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 12731–12750, Nov. 2022, doi: 10.18653/v1/2023.findings-acl.806.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,” *EMNLP 2018 - 2018 EMNLP Work. BlackboxNLP Anal. Interpret. Neural Networks NLP, Proc. 1st Work.*, pp. 353–355, Apr. 2018, doi: 10.18653/v1/w18-5446.

[24] M. Chen *et al.*, “EfficientQAT: Efficient Quantization-Aware Training for Large Language Models,” Jul. 2024, Accessed: Feb. 20, 2025. [Online]. Available:

https://arxiv.org/abs/2407.11062v2.

[25] A. Wang *et al.*, “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems,” *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[26] B. Wang *et al.*, “Adversarial GLUE: A Multi-Task Benchmark for Robustness Evaluation of Language Models,” Nov. 2021, Accessed: Feb. 20, 2025. [Online]. Available: https://arxiv.org/abs/2111.02840v2.