From BERT to Mamba: Evaluating Deep Learning for Efficient QA Systems

Eli Guo
Center for Data Science
New York University
yg3030@nyu.edu

Anirudh Garg
Courant Institute of
Mathematical Sciences
New York University
ag9563@nyu.edu

Shubham Goel
Tandon School of Engineering
New York University
sg4599@nyu.edu

Anjel Patel
Tandon School of Engineering
New York University
ap8589@nyu.edu

Navya Kriti
Tandon School of Engineering
New York University
nk3696@nyu.edu

Abstract

This project aims to compare the performance of multiple deep learning models for a Question Answering NLP task, focusing on balancing accuracy and computational efficiency. Effective question-answering requires models to interpret queries precisely within a given context while remaining resource-efficient for real-world applications. We fine-tune, and evaluate various deep learning models such as BERT[1], T5[2], and Mamba[3], assessing their Exact Match scores and resource usage. Through this comparative analysis, we explore the trade-offs that shape each model's performance, providing insights into optimizing QA systems for practical deployment. Code is available at: https://github.com/eliguo/DS-GA-1011-Group-Project.

1 Introduction

Question Answering is an essential NLP task requiring models to interpret queries and extract information with precision. In this project, we fine-tuned and evaluated BERT, T5, LSTMs, and Mamba on QA tasks using the SQuAD dataset. While LSTMs effectively captured local sequential patterns, they struggled with long-range dependencies. Transformer-based models like BERT and T5 addressed these limitations but posed challenges due to high computational demands. Mamba, optimized for efficient long-sequence processing, was also explored. Our analysis compared Exact Match scores and memory usage, providing insights into balancing accuracy and efficiency in QA systems.

2 Related Work

The BERT model, introduced by Devlin et al. (2019), uses bidirectional Transformer layers to capture complex word dependencies, achieving state-of-the-art results on QA benchmarks like SQuAD [4], though its high memory usage and limited input length present challenges in resource-constrained settings. T5, developed by Raffel et al. (2020) [2], reformulates NLP tasks as text-to-text, offering flexibility across applications, while earlier models like LSTMs [5] effectively captured sequential patterns but struggled with long-range dependencies. The Mamba model, introduced by Gu and Dao (2023), utilizes selective state space models (SSMs) for efficient, linear scaling without

attention layers, enabling the processing of sequences up to one million tokens. Although Mamba holds potential as a scalable solution for sequence modeling, its performance in QA tasks remains unexplored.

3 Approach

BERT Model. We fine-tuned the BERT-Base-Uncased model on the SQuAD dataset for question-answering tasks by training on question-answer-context pairs with specific hyperparameters to optimize Exact Match (EM) scores. While effective, this process was computationally intensive due to BERT's high memory and resource demands.

To improve efficiency, we applied Static Model Pruning (SMP) [6], which removes unimportant weights without fine-tuning by using a binary mask to retain only critical weights. This significantly reduced the parameter count while maintaining performance. SMP leverages first-order information for pruning, achieving better task alignment and efficiency at various sparsity levels.

Additionally, we incorporated Knowledge Distillation [7], aligning the pruned model's predictions with those of a fine-tuned BERT model on SQuAD. This approach enhanced accuracy despite reduced parameters. Together, these methods were able to achieve strong performance at high-sparsity

T5 Model. We fine-tune the T5-base model from Hugging Face for QA tasks, preparing data from JSON files similar to SQuAD to extract question, answer, and context pairs. The T5 tokenizer encodes these inputs with maximum lengths for questions (Q_LEN) and answers (T_LEN), using padding and truncation as needed. Data is structured into a custom `QA_Dataset` class, with data loaders managing batch processing for training and validation. Fine-tuning on GPU uses the Adam optimizer, tracking performance via training and validation loss, and saving the best checkpoint based on validation loss to ensure effective answer generation.

The fine-tuning process took 4 hours for 3 epochs, making it computationally expensive. To optimize resources, we explored LoRA (Low-Rank Adaptation) [8] and quantization [9]. LoRA reduces trainable parameters by adding lightweight adapters, enabling efficient fine-tuning without significant accuracy loss. Quantization compresses the model by using lower precision formats (e.g., INT8), reducing size and improving inference speed, with minor accuracy trade-offs.

Additionally, we experimented with combining LoRA and quantization to leverage their complementary benefits. This hybrid approach (QLoRA[10]) achieves efficient fine-tuning by training a small subset of parameters while maintaining inference efficiency and reduced memory footprints from quantization. Performance metrics (exact match), are evaluated for each configuration to balance accuracy, model size, and computational efficiency.

LSTM-based Model. The model processes context and question inputs through embeddings and bidirectional LSTMs to capture context. An attention mechanism highlights relevant parts, followed by concatenation, an LSTM, flattening, dropout for regularization, and dense layers to predict answer start and end positions.

To enhance performance, we employed pre-trained embeddings, attention mechanisms, and regularization. GloVe captures global word-level semantics, while FastText adds subword-level details, improving handling of rare words. Multi-Head Attention focuses on multiple sequence aspects, while Self-Attention efficiently identifies relevant parts within a sequence, making it suitable for medium-scale tasks. Bidirectional LSTMs capture context from both directions, and Residual Connections preserve information and improve gradient flow, enhancing learning. Global MaxPooling efficiently extracts key features, and L2 regularization with Dropout prevents overfitting, ensuring better generalization. With self-attention, higher performance was achieved.

Mamba Model. For the Mamba model, we utilized the `state-spaces/mamba-130m` variant, a compact model with 130 million parameters. To adapt Mamba for QA, we employed a two-step approach. Initial experiments utilized the pre-trained model with basic n-shot prompting to elicit answers to trivia-style questions. By constructing examples within the prompt, the model was guided toward structured response generation. However, this approach demonstrated limited accuracy, motivating the need for fine-tuning.

Subsequent experiments fine-tuned the Mamba model on the SQuAD v2.0 dataset [4] using supervised learning to predict answer spans for answerable questions and "no answer" for unanswerable ones. Data augmentation paired questions with unrelated contexts to distinguish between answerable and unanswerable queries effectively. Cross-entropy loss optimized predictions while addressing sequence-length and memory constraints.

4 Experiments

4.1 Data

Our dataset, SQuAD 2.0 [4], is a widely used QA benchmark that builds on the original SQuAD with over 50k unanswerable and 100k answerable questions, challenging models to provide accurate answers or recognize when a question cannot be answered from the given context. Each instance includes a context paragraph, a question, and either an answer span or an indication of unanswerability.

4.2 Evaluation Method

Our main evaluation metric is Exact Match, a standard for assessing answer quality in QA tasks, especially for span-based systems. EM calculates the percentage of predictions that exactly match ground-truth answers in SQuAD, requiring identical text, punctuation, capitalization, and whitespace.

4.3 Experimental details

BERT Model. In our experiments, we fine-tuned the model over 5 epochs. We used a learning rate of 0.00001 along with AdamW optimizer. During training, BERT's built-in loss function for QA tasks was used, which calculates the combined loss from the predicted start and end positions of the answer spans.On A100 GPU, each epoch took 50 minutes of training + 2 minutes of evaluation.

To explore efficiency, we extended our experiments to include pruning and knowledge distillation :

- **SMP (Static Model Pruning:** We applied SMP with 10% remaining weights, resulting in 85M parameters due to the encoder size and binary mask representation. This method was run for 10 epochs, significantly faster than fine-tuning, and achieved strong results while improving computational efficiency.
- **Static Model Pruning + Knowledge Distillation:** Using KD, we retained 50% of the weights, keeping 85M parameters because of the binary mask overhead. This approach, also run for 10 epochs, provided improved performance compared to fine-tuning.

T5 Model. In our T5 experiments, we fine-tuned the model for 5 epochs using a 0.00001 learning rate with the Adam optimizer, leveraging T5's QA loss function based on predicted answer spans. Each epoch took 75 minutes for training and 15 minutes for evaluation.

To further explore efficiency and scalability, we extended our experiments to include LoRA, quantization, and QLoRA methodologies:

- **LoRA (Low-Rank Adaptation):** By fine-tuning only a small number of lightweight adapters added to specific layers, training with LoRA required just 30 minutes per epoch due to the reduced number of trainable parameters, making it significantly faster while maintaining competitive performance.
- **Quantization:** Applying post-training quantization reduced the model size substantially, compressing it from 850 MB to approximately 212 MB in INT8 precision. This enabled faster inference times, though with a minor trade-off in accuracy. Combined training and evaluation times for quantized models remained close to the original, as quantization primarily impacts inference efficiency.
- **QLoRA:** Combining the benefits of LoRA and quantization, QLoRA further minimized computational requirements during fine-tuning while leveraging quantized weights for memory efficiency. Each epoch in QLoRA took approximately 25 minutes, a further reduction from LoRA alone, and the model size remained compressed at around 220 MB.

LSTM-based Model. In the LSTM experiment, we trained **first model** on 10,000 samples with 2,000 for validation, as each epoch took over 4 hours on a T4 GPU. We used batch size of 8, Adam optimizer with learning rate of 0.001, categorical cross entropy loss, calculating loss based on predicted start and end positions. The model was trained for 10 epochs.

The model showed poor generalization performance indicating a lack of generalization. To improve performance while considering computational efficiency, various scenarios were tested. In some cases, training beyond 11 epochs led to overfitting. Accordingly, following scenarios were explored:

- **Second Model** was trained for 11 epochs with batch size of 32 on A100 GPU which took around 10 minutes. 20,000 training samples and 5,000 validation samples were used. GloVe embeddings, L2 regularization, Dropout, Basic attention and Flatten pooling were used, limiting its contextual understanding. The results indicate overfitting and a need for architectural improvements.

- **Third model** was trained for 8 epochs with batch size of 16 on A100 GPU which took around 36 minutes including evaluation. 60,000 training samples and full validation set (11,873 samples) were used. It leverages pre-trained FastText, Multi-Head Attention, GlobalMaxPooling1D and uses residual connections to preserve information across layers. Better generalization was realized.

- **Fourth model** was trained for 8 epochs with batch size of 16 on A100 GPU which took 28 minutes including evaluation. The same data set size as for the third model was used. It utilizes Self-Attention, FastText embeddings, residual connections, and GlobalMaxPooling1D, which leads to a significant performance boost.

Details about the models are mentioned in Table 3 for reference. More detailed table can be accessed at: `https://github.com/eliguo/DS-GA-1011-Group-Project/blob/main/LSTM/readme.md`.

Mamba Model. The smallest variant of Mamba (130M parameters) was fine-tuned for 10 epochs using an AdamW optimizer with a learning rate of 1×10^{-5}. Training was conducted on an A100 40GB GPU, with each epoch requiring approximately 22 minutes. The dataset was preprocessed to include both positive and synthetic negative samples, where questions were paired with unrelated contexts to train the model to differentiate between correct answers and unanswerable queries. This fine-tuning approach enabled the model to output "I don't know" for unanswerable cases.

Despite these efforts, the model achieved an EM score of 0.12, far below state-of-the-art QA models, highlighting Mamba's limitations in semantic reasoning and context comprehension. Scalability issues with larger variants and challenges with long input sequences added to the difficulties despite its theoretical efficiency.

4.4 Results

BERT and T5 outperform LSTM and Mamba due to superior architecture but are resource-intensive. We explored model compression techniques like pruning for more practical QA systems

Model	Remaining Weights	Model Size	Accuracy (EM)
Bert-Base-Uncased + Fine Tuning	100%	∼110M	0.65
Static Model Pruning	10%	∼85M	0.75
Static Model Pruning + KD	50%	∼85M	0.82

Table 1: Comparison of BERT Model Variants

Model	Parameters	Trainable Parameters	Model Size	Accuracy (EM)
T5	220M	220M (100%)	850MB	0.74
T5 + LoRA	220M + 800K	800K (0.3%)	850MB	0.71
T5 + Quantization	220M	220M (100%)	212MB	0.72
T5 + QLoRA	220M + 800K	800K (0.3%)	212MB	0.69

Table 2: Comparison of T5 Model Variants

Model	Model Parameters	Model Size	Accuracy (EM)
Model_1	499M	1.86GB	0.094 (Training), 0.008 (Validation)
Model_2	51M	195MB	0.102 (Training), 0.029 (Validation)
Model_3	36M	138MB	0.071 (Training), 0.116 (Validation)
Model_4	36M	138MB	0.972 (Training), 0.220 (Validation)

Table 3: Comparison of LSTM models

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[3] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[4] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[6] Ting Jiang, Deqing Wang, Fuzhen Zhuang, Ruobing Xie, and Feng Xia. Pruning pre-trained language models without fine-tuning. *arXiv preprint arXiv:2210.06210*, 2023.

[7] Yasaman Boreshban, Seyed Morteza Mirbostani, Gholamreza Ghassem-Sani, Seyed Abolghasem Mirroshandel, and Shahin Amiriparian. Improving question answering performance using knowledge distillation and active learning. *arXiv preprint arXiv:2109.12062*, 2021.

[8] Phillip Wallis Zeyuan Allen-Zhu Yuanzhi Li Shean Wang Lu Wang Weizhu Chen Edward J. Hu, Yelong Shen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[9] Deepthi Karkada Vivek Menon Sun Choi Kushal Datta Vikram Saletore Aishwarya Bhandare, Vamsi Sripathi. Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv preprint arXiv:1906.00532*, 2019.

[10] Tim Dettmers · Artidoro Pagnoni · Ari Holtzman · Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *https://neurips.cc/virtual/2023/poster/71815*, 2023.