Shallower LLMs for Faster Inference

Ronit R. Arora
Department of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
ronitarora@gatech.edu

Nathan D. Wang
Department of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
nwang334@gatech.edu

Uzair Akbar
Department of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
uzair.akbar@gatech.edu

Abstract

Today's LLMs have billions of parameters contained in a handful of large transformer blocks. While these transformers are highly accurate, computational costs stemming from the sheer number of parameters to utilize in inference alongside large transformer depth can lead to increased inference latency. Due to the auto-regressive nature of LLMs, this latency can build up linearly over time. We seek to reduce the inference time of LLMs through architectural changes. In our new architecture based upon Llama from Meta, we propose enriching the transformer block architecture - adding a nonlinear layer on top of each attention head before concatenation, while reducing the total number of transformer blocks. We demonstrate[1] our proposed models have reduced inference latency and maintain or improve upon evaluation results compared to larger original models, revealing the potential to replace larger LLMs with richer, shallower LLMs .

1 Introduction

Large Language Models such as GPT-175B, PaLM-2, and Claude 2 have gained immense popularity, serving millions of users daily with their conversational generations. For example, ChatGPT alone has a weekly user base of 100 million, generating a vast volume of tokens [1]. Considering the auto-regressive nature of the generation of tokens through large models with billions of parameters, the decoding time per token becomes a significant concern. The cumulative effect of the auto-regressive generation results in potential queuing delays and increased time overhead for users. Therefore, we aim to mitigate inference latency for user experience.

There are two major difficulties when attempting to solve the problem of inference latency. The first stems from the colossal size of these models, which leads to other limitations for research. The model we work with for parameter reduction is Meta's most recent Llama2 architecture. Llama's largest 65 billion-parameter model presents substantial computational and time-limitation challenges, and performing queries or training tasks with such models on standard devices may be unfeasible. We employ a single Nvidia A100 GPU for our modified Llama training and evaluation pipelines.

The second major obstacle faced is maintaining model accuracy after model compression. While state of the art solutions exist and aim to reduce inference time latency, techniques like quantization

[1]Private GitHub repository link: https://github.gatech.edu/cs8803smr-f23/research14

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

and pruning often compromise model accuracy. This compromise is evident in output sequences with noticeably higher perplexity, a common Large Language Model performance metric.

While existing system-level optimizations like KV-caching and speculative decoding exhibit promising advancements, their limitations to post-trained models hinder widespread accessibility and comprehensive reduction in inference latency during training. Additionally, Nvidia's TensorRT LLM which combines many such optimizations, is a notable accelerator to reduce inference latency during the decode phase. However, its exclusivity to highly esoteric, expensive Nvidia chips limits widespread accessibility. Despite these system-level enhancements, the question remains: can we further reduce inference latency through architectural modifications while maintaining model accuracy across common performance metrics?

We therefore propose a novel method: the creation of richer, denser attention heads with linear and non-linear Swish layers between heads in order to maintain model accuracy while reducing the depth of the network. We reduce the overall number of model parameters by dropping transformer blocks, but we achieve similar or better accuracies by adding differentiable, learnable layers on top of attention heads.

2 Related Work

With regard to the aforementioned issue with inference latency, there has been significant interest in compressing previously trained models or employing speculative execution techniques to more efficiently decode for inference [17]. These post-training approaches encompass techniques such as pruning less useful weights [5, 16, 11], quantization [7], knowledge distillation [18], and low-rank factorization [11]. In many cases, these methods compromise model accuracy, and therefore we elected to focus on bounding our models prior to training. Beyond methods like quantization aware training, there has been less research concerning the reduction of parameters during training. By reducing parameters either before or during training, the model acquires a more streamlined and effective representation to the tasks at hand. This has the potential to result in improved performance, quicker convergence, and decreased overfitting when contrasted with the practice of training a larger model and compressing it post-training.

3 Proposal

For the purpose of our experiments, we leveraged Meta's recently published open source large language mode, Llama2. This architecture integrates state of the art methodologies aimed at enhancing overall generalization and model learning capabilities [8]. For example, Llama2 incorporates RMSNorm before segmenting into the query, keys, and values matrices and applies Rotary Positional Embeddings to achieve an optimal fusion of absolute and relative approaches in positional embeddings. Furthermore, Llama's feedforward block uses the SwiGlu activation function, which is a combination of Google's Swish [13] and the Gated Linear Units activation functions [12]. In comparison to other activation functions like ReLU, Swish has a smoother, non-monotonic curve, which allows for better optimization and faster convergence. Consequently, we opted to apply the self-gated Swish function as our non-linear activation function between heads.

In line with Google's research, which experimented with depth-wise convolutional layers applied to the Q, K, and V matrices, our objective is to modify state of the art transformer models for more effective learning through shallower architectures [5]. Given the computationally expensive and increased parameter requirements associated with convolutional layers, we opted to employ fully connected linear and non-linear layers and evaluate across accuracy and inference time. Incorporating non-linearity between heads enables the model to learn more complex patterns between tokens. Our proposed method further diverges from So et al. [4] in terms of layer placement. While their approach adds layers onto QKV projections, we propose adding layers between each head following the attention score calculations. This difference aims to constrain the number of additional parameters, striking a balance between computational efficiency and expressive capability of our novel model.

Furthermore, prior research on convolutional and deep neural networks with regards to the impact of wider rather than deeper networks exhibited promising findings for our similar project in reducing the depth of large language models while increasing the complexity of the attention head mechanism [6]. Brown et al. demonstrated that the integration of a single, broader attention layer within each

transformer block led to enhanced model performance compared to deeper baseline models [9]. This modification not only improved the model's effectiveness but also reduced inference latency and memory demands. Building upon this insight, we have refined our approach by opting for a richer attention mechanism, which incorporates linear and non-linear layers between heads, instead of wider layers. This modification will further enhance the model's capacity to capture more complex relationships between tokens, potentially yielding superior performance compared to only employing broader attention layers.

3.1 Architecture Overview

For a transformer block from Llama's architecture, we propose to add a linear layer followed by a sigmoid linear unit (SiLU) activation on each attention head before concatenation. Inside the transformer block, each attention head (not including batch size) is of shape $(s,\ d/n)$, where s indicates max sequence length, d indicates hidden dimension, and n indicates the number of attention heads in a block. Our proposed linear layer is of size $(d/n, d/n)$. We create a linear layer for each of the heads, and perform a matrix multiplication of the head with its corresponding linear layer. We additionally add d/n bias values, and follow up with our SiLU non-linearity. In essence, this performs a nonlinear operation across the rows of the attention head.

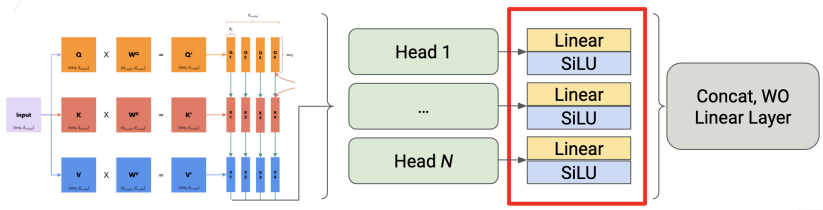


Figure 1: Proposed architecture design, with our methods highlighted inside a red box. We utilize and modify the transformer block from Meta's Llama model. Instead of adding parameters directly after K, V, Q, we add layers onto the attention heads before concatenation of the heads. We adopt the left-most diagram from [19].

3.2 Reasoning

Through our enriching of rows only across the attention head with a nonlinear layer, we add a minimal number of parameters relative to the size of the transformer block. Our proposal adds a total of:

$$\left(\underbrace{\frac{d}{n} * \frac{d}{n}}_{\textbf{Layer size}} + \underbrace{\frac{d}{n}}_{\textbf{Bias}} \right) * \underbrace{n}_{\textbf{Per head}} = \frac{d^2}{n} + d$$

parameters per transformer block. If otherwise performed as a fully connected layer on each attention head, the total number of parameters would be extremely large as our input and output size would scale by a factor of the sequence size (first dimension of the attention head), which is not feasible. We add layers per attention head instead of after concatenation with similar logic, as otherwise this would introduce too many extraneous parameters across all attention heads, scaling the parameter size approximately by n^2 (e.g. if a linear layer was added after concatenation, it would be of size (d, d)). Through our specific addition of layers on the attention heads, we maintain original dimensions before and after - i.e. the attention head before and after the layer multiplication is $(s, d/n)$. By maintaining these dimensions, we reduce the effect of changing any other components of the model to make the experiment controlled.

3.3 Implications Overview

Our reasoning behind this architecture decision is to create richer transformer blocks that will allow for reducing the overall number of transformer blocks utilized in the model. Consequently, the depth of the model can be significantly decreased compared to its original form while utilizing the deeper

transformer blocks to compensate for lost accuracy. Through this overall net reduction in depth, we aim to reduce the effective inference time for the model. To measure the effective change in latency and accuracy, we train 8 different models of different sizes - 4 baseline models and 4 modified models - ranging from 5 through 8 transformer blocks inclusive. We then provide a complete comparison in our evaluation to shed more light on the latency implications. In Section 3.4 we provide more details of our procedure in training these 8 models.

3.4 Model Training and Evaluation Procedures

In order to assess the impact of our proposed modifications on large language models such as Llama2 across common language metrics and inference latency, we devised the following experimental pipeline:

Setup for Retraining Llama2: We initiated by adapting Meta's Llama repository inference-only code [10], where inference model architecture is defined. We modify this code to remove KV-caching for training, since KV-caching is for inference rather than the training process. Importantly, we reduce the number of parameters in order to perform a controlled experiment to compare against our proposed models and train all models in a feasible amount of time. We reduce s (sequence length) from 2056 to 512, d (hidden dimension) from 4096 to 1024, n from 32 to 8, and scale down SwiGLU hidden layer size by a factor of 4. Our parameter counts can be found below in Table 1.

	5 Layers	**6 Layers**	**7 Layers**	**8 Layers**
Baseline Model Param. Count	128,789,504	141,440,000	154,090,496	166,740,992
Proposed Model Param. Count	129,449,984	142,232,576	155,015,168	167,797,760
# of New Param. Added	660,480	792,576	924,672	1,056,768

Table 1: The parameter counts of each of our eight models. We do not add a large number of parameters and still demonstrate significant improvements in our evaluation.

For our training dataset, we elected to use a subset of the SlimPajama dataset [3], which is a cleaned subset of the expansive RedPajama dataset. The RedPajama dataset was created to heavily mimic the dataset Meta used to train their original Llama models [14]. SlimPajama incorporates data from Common Crawl, Stack Overflow discussions, literary works in the form of novels, as well as question and answer forums and various other content sources. Lastly, we utilize the pretrained Llama tokenizer for encoding and decoding of texts/tokens.

Modification of Llama2 Model: Prior to implementing our proposed model changes, we modified Meta's Llama2 model file and appended a standard training loop for comprehensive retraining and tracking training metrics. In the original architecture of the Llama2 model, the forward pass initially applies the RMSNorm to the input data. Following this, linear transformations are applied individually to the QKV projections. Rotary positional embeddings and the computation of attention scores are subsequently utilized to generated the weighted representations based on the values matrix. Implementing our proposed modifications, we introduce linear and non-linear SiLU layers after each attention head at this particular stage of the model's forward pass, visually depicted in Figure 1. This aims to promote further learning within the attention heads, while importantly maintaining the original dimensions prior to the conventional concatenation across attention heads and final W_0 linear transformation.

Distributed Processing Setup: To ensure distributed processing capabilities and parallelization, in line with the techniques employed in the original Llama2 model, we integrated a matrix-based approach to apply a fully connected layer and biases to each attention head independently. Our training and evaluation scripts were run on an Nvidia A100 GPU.

Model Configuration and Training: Based on initial experiments considering model size, memory demands, and time constraints, we opted to evaluate models with varying depths (5, 6, 7, and 8 transformer blocks). We utilize a cross-entropy loss function on output model logits, similar to Llama. We created scripts for standardized model configurations with the specified arguments (batch size, head dimensions, sequence length, number of blocks, and number of heads) for a more streamlined, efficient training process. Following established best practices,

we implemented checkpointing, storing of training metrics, and additional logs for managing our training sessions. In our experimentation, we conducted training for one epoch using a dataset comprising of 1.5 million datapoints. We utilize this smaller amount of data to expedite training of all models, which already can take nearly a full day with our setup. Our critical assumption here is that we can effectively compare undersatured baseline and proposed models, since saturating models fully is infeasible due to the millions of GPU hours required. We note this assumption in subsection 5.1. Despite each model remaining unsaturated, our proposed modifications were expected to demonstrate noticeable effects when compared to the baseline models.

Evaluation Metrics and Testing: We modified Llama2's generation script for token generation to fit our evaluation metric and dataset requirements. The evaluation encompassed chat and sentence completion tasks, as well as a wiki question and answer task, measuring performance using common language metrics like BLEU, Rouge, and perplexity. We conducted evaluations using both in-distribution (SlimPajama) and out-of-distribution (Anthropic [2] and WikiQA) datasets, generating responses and scoring them based on the language performance metrics selected. Tabulated scores were calculated by mean, median, 5th percentile, 95th percentile, and standard deviation per test dataset and output to a csv file for further analysis. For some parts of this evaluation, we adapt the TRACE evaluation framework [15].

Inference Time Evaluation: For evaluating inference time, we averaged the runtimes of prompt responses in nanoseconds during the token generation loop in our evaluation script. We converted inference times to microseconds to enhance the visibility of distinctions and trends across our different models.

Visualization of Results: Figures and plots were created to visually represent and highlight the trends observed across our models based on the evaluated language metrics and inference times.

This pipeline expedited a comprehensive exploration of our proposed architectural modifications' impact on Llama2's performance in terms of language understanding and inference latency.

4 Experiments & Evaluation

4.1 Evaluation Metrics

We employ the following widely-used LLM evaluation metrics to measure the quality of our generated text in comparison to our test dataset reference text:

BLEU-n: Bilingual Evaluation Understudy (BLEU) involves a 0 to 1 n-gram precision score between the generated texts and the references, considering how many and the length of n-grams in the generated text overlap with those of the references. Despite its simplicity and wide acceptance, BLEU cannot capture semantic or contextual understanding, will not account for grammar or coherence, and might penalize different word orders and synonyms.

ROUGE: Recall-Oriented Understudy for Gisting Evaluation (ROUGE) measures the overlap and similarity between the n-grams in the generated summary and the reference. It can evaluate overlap, longest common subsequence, and weighted n-grams of text summarization generations. Similar to BLEU, ROUGE cannot capture semantic or contextual understanding, is sensitive to length of generation, and may reward redundancy.

FuzzyWuzzy: FuzzyWuzzy measures the similarity between two strings based on their character-level differences, typically employing Levenshtein distances or Ratcliff-Obershelp similarity. We use this metric as an additional evaluation for our text generations, but it has the same shortcomings as BLEU.

Perplexity: Perplexity measures how well an LLM predicts a test dataset by assessing the model's confidence when predicting the next word in a sequence of words. It is calculated as $2^{cross-entropy}$ and allows for a quantitative evaluation of model performance. Just like the other metrics, it struggles with evaluating semantic and coherence understanding.

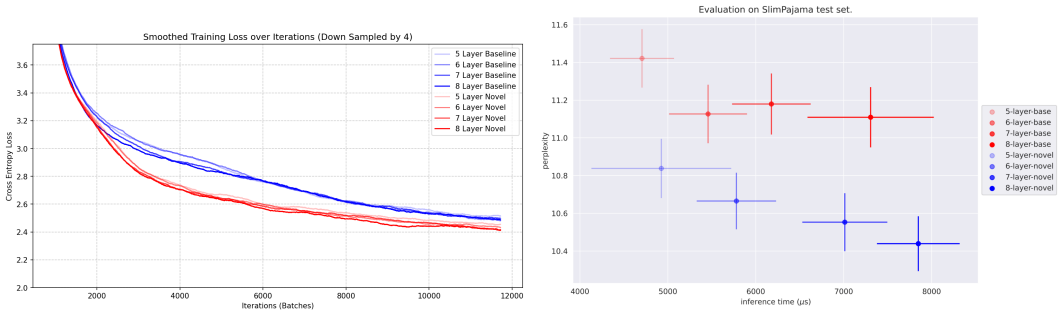
The most prominant issue with these metrics is that a better score does not necessarily correlate with better performance in real-world applications, since understanding context and generating coherent text are crucial. Nevertheless, these metrics can provide quantitative insight on the performance of LLM models and enables us to perform a comparative analysis on our novel models with regard to their respective baselines.

4.2 Results

We make a distinction between evaluating our model on a test dataset with the same source as the training dataset, i.e. an *in-sample* evaluation as opposed to evaluation on test datasets that were collected in a completely different way, i.e. *out-of-distribution* evaluation.

Given the limited training time, dataset size and model size, we believe that such a distinction is important and the in-sample results would be most representative of the performance of our model. Nevertheless, we also include out-of-distribution results for completeness purposes.

4.2.1 In-sample Evaluation



(a) Loss on the SlimPajama training dataset.

(b) Perplixity on the SlimPajama test dataset.

Figure 2: We see a consistent improvement for all novel models compared to their respective baselines across both training and test sets while at the same time having lower inference time compared to baselines of higher depth. Furthermore, we also see a trend towards both worse training loss and test perplexity with decreased number of layers and a significant reduction in inference time as well. Note for (a), we perform moving average smoothing of 1000 iterations, which are previously down-sampled. During our training, we logged loss values every fourth iteration/batch.

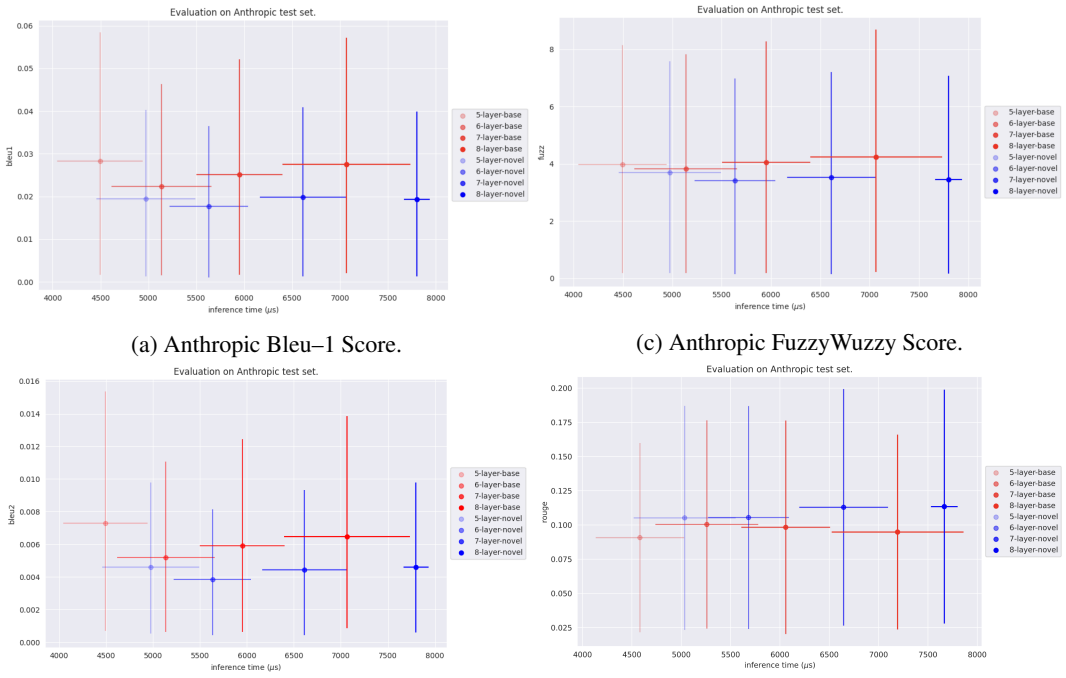
We cite two results presented Fig. 2 for our in-sample evaluation on the SlimPajama dataset. The training-loss in Fig. 2a shows a consistent divergence between the baselines and the novel models, implying that the novel models are able to fit the training data much better given the same amount training time.

Furthermore, Fig. 2b shows that each novel model is not only significantly faster in terms of inference time, but also has better perplexity compared to deeper baselines.

4.2.2 Out-of-Distribution Evaluation

We chose the Anthropic dataset for out-of-distribution evaluation of our models. The assumption being that given a large enough training dataset, out models should be able to generalise reasonably well to unseen datasets as well. However, these results need to take more as a motivation rather than a final verdict because of the limitations of our training pipeline (smaller models to fit on a single GPU, smaller dataset, shorter training times, etc.). We run a battery of tests with different evaluation metrics as shown in Fig. 3.

Of note here is the significant decrease in novel model inference times over deeper baselines while not having a significant decrease in any evaluation metric. While this is encouraging as an initial result, we should also note that no significant difference can also be seen with the addition of more layers in either novel or baseline variants. This implies that the above stated limitations of the training pipeline are indeed a bottleneck for achieving more conclusive results.



(a) Anthropic Bleu–1 Score.

(c) Anthropic FuzzyWuzzy Score.

(b) Anthropic Bleu–2 Score.

(d) Anthropic Rouge Score.

Figure 3: Out-of-Distribution evaluation on the Anthropic dataset (chat completion). We see a consistent and sometimes significant improvement of each novel model on inference time compared to deeper baselines while the evaluation metric does not decrease significantly. It should be noted however that while these results are encouraging, they should be explored more thoroughly for a final verdict.

Additional out-of-distribution results are provided in Appendix A.

5 Discussion

5.1 Assumptions and Limitations

One limitation of our analysis is the size of our models and dataset. We attempt to maintain as large a transformer as possible to best represent an LLM. Our largest trained model has nearly 170 million parameters. Our dataset is somewhat small with 1.5 million text datapoints, although much cleaner given we are using a processed/cleaned version of RedPajama. We cannot feasibly train a llama-sized model to saturation without a cluster of GPUs for a long period of time, which leads to our second assumption: We assume that comparisons of under-saturated baselines and proposed models are still somewhat valid based upon initial loss convergence metrics and task-specific metrics. For future work and with longer timelines, these assumptions may lead us to train larger models for longer periods of times to further validate our hypothesis.

5.2 Conclusion

The autoregressive nature of token generation poses substantial inference latency concerns, impacting user experience. Previous research in determining methods to diminish inference latency in large language models reveals critical challenges in maintaining model accuracy while reducing inference latency. Our focus on Meta's Llama2 architecture, particularly with regard to parameter reduction, emphasizes the complexity of handling massive models due system and hardware constraints.

The inherent trade-off between model compression and model accuracy has been a persistent challenge, as state of the art compression techniques like quantization and pruning often

compromise model accuracy, leading to a less accurate understanding of language.

Our study explores architectural modifications to reduce inference latency while preserving model accuracy. Specifically, our novel approach introduces richer attention heads with inserted linear and non-linear SiLU layers between heads to enhance the network's learning with shallower, parameter-reduced architectures.

Upon experimentation, our findings revealed a compelling trend: our novel 5-layer model performed comparably to the 8-layer baseline. Intriguingly, there was a consistent improvement in perplexity with our novel models compared to their respective baselines. More importantly, our novel models have been shown to perform similarly or better than deeper baseline models. Moreover, our novel models showcased faster convergence in terms of training loss, emphasizing their efficiency in learning complex relationships between tokens.

In conclusion, our experimentation presents a novel approach that balances reduced network depth with maintained model accuracy via learnable layers between heads. The observed trend of comparable performance between the novel 5-layer model and the deeper 8-layer baseline, expresses the viability of our proposed architectural modifications. We wish to further experiment with this approach by adding additional non-linearities and training for longer on larger datasets, based upon the limitations discussed previously. Still, these promising results illustrate the potential of our approach in advancing large language models for improved inference efficiency without compromising language performance metrics.

6 Statement of Collaborator Contribution

6.1 Ronit Arora's Contributions (RA, NW, UA)

Ronit developed all code files related to training the baseline and proposed models. Ronit worked on the theory for adding additional layers, adapted Llama's models and modified them with proposed layers. Ronit created the training pipelines, set up the distributed environment, and managed and ran all training jobs. He created the frameworks for obtaining all relevant training metrics, including losses, times, and more. Ronit additionally analyzed and visualized training losses and developed the perplexity evaluation plus corresponding visualizations. He helped maintain the repositories and GPU setup on the remote VM. Lastly, he contributed to all written works, from the proposal to midpoint and final presentations and reports.

6.2 Nathan Wang's Contributions (RA, NW, UA)

Nathan helped to adapt the models for training by identifying parts of the code utilizing the KV-cache and modifying such. Nathan helped to set up the necessary drivers and GPU setup on the remote VM. He furthermore helped to develop some of the visualizations of our evaluation and run evaluation jobs for obtaining outputs. He additionally contributed to all written works to a high extent.

6.3 Uzair Akbar's Contributions (RA, NW, UA)

Uzair wrote the implementation of sentence-generation and chat-completion scripts for our models for use in our evaluation pipeline. He also adapted the TRACE evaluation framework and created evaluation scripts. He ran all trained models across multiple out-of-distribution datasets (i.e. outside of SlimPajama) to obtain evaluation metrics such as BLEU, Rouge, and more. He helped measure the inference latencies of our models and visualized metrics. Lastly, he also contributed to all written and presentational works.

References

[1] Aisha Malik, "OpenAI's ChatGPT now has 100 million weekly active users," TechCrunch, Nov. 06, 2023. https://techcrunch.com/2023/11/06/openais-chatgpt-now-has-100-million-weekly-active-users.

[2] "Anthropic/hh-rlhf · Datasets at Hugging Face," huggingface.co, Dec. 06, 2023. https://huggingface.co/datasets/Anthropic/hh-rlhf (accessed Dec. 11, 2023).

[3] Daria Soboleva, "SlimPajama: A 627B token, cleaned and deduplicated version of RedPajama," Cerebras, Jun. 09, 2023. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama (accessed Dec. 10, 2023).

[4] David R. So., et al., "Primer: Searching for Efficient Transformers for Language Modeling," CoRR, 2021.

[5] Elias Frantar, Dan Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," CoRR, 2023.

[6] Eran Malach, Shai Shalev-Shwartz, "Is Deeper Better only when Shallow is Good?," CoRR, 2019.

[7] Guangxuan Xiao, et al., "SmoothQuant: Accurate and efficient post-training quantization for large language models," CoRR, 2022.

[8] Hugo Touvron, et al., "LLaMA: Open and Efficient Foundation Language Models," CoRR, 2023.

[9] Jason Ross Brown, et al., "Wide Attention Is The Way Forward For Transformers?," CoRR, 2022.

[10] Meta, “Llama 2,” Meta AI. https://ai.meta.com/llama, (accessed Dec. 10, 2023)

[11] Mingyang Zhang, et al., "Pruning meets low-rank parameter-efficient fine-tuning," CoRR, 2023.

[12] Noam Shazeer, "GLU Variants Improve Transformer," CoRR, 2020.

[13] Prajit Ramachandran, et al., "Searching for Activation Functions," CoRR, 2017.

[14] "RedPajama, a project to create leading open-source models, starts by reproducing LLaMA training dataset of over 1.2 trillion tokens," www.together.ai. https://www.together.ai/blog/redpajama (accessed Dec. 10, 2023).

[15] Xiao Wang, et al. "TRACE: A Comprehensive Benchmark for Continual Learning in Large Language Models," arXiv:2310.06762 [cs.CL], 2023.

[16] Xinyin Ma, et al., "LLM-pruner: On the structural pruning of large language models," CoRR, 2023.

[17] Xunyu Zhu, et al.,"A Survey on Model Compression for Large Language Models," CoRR, 2023.

[18] Yuxian Gu, et al., "Knowledge distillation of large language models," CoRR, 2023.

Additional Diagram Reference:

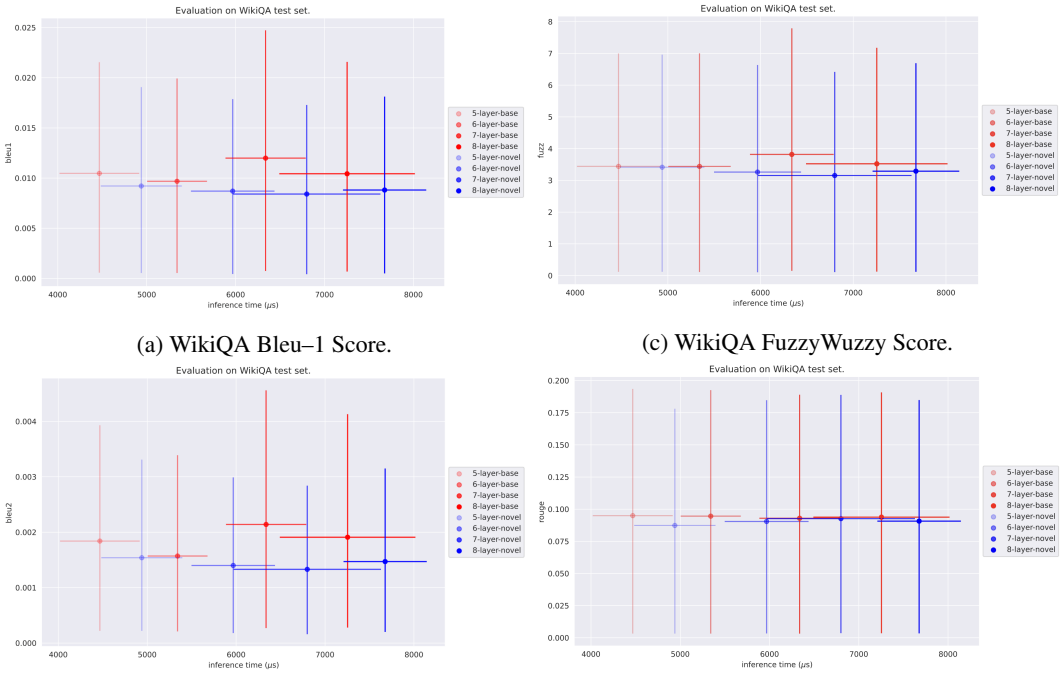
[19] Umar Jamil. "LLaMA explained: KV-Cache, Rotary Positional Embedding, RMS Norm, Grouped Query Attention, SwiGLU" [Video]. YouTube, 2023. https://www.youtube.com/watch?v=Ya_4rht4DG8

A Additional Evaluation Results

We also evaluate on the WikiQA dataset as an additional out-of-evaluation result. The results are presented in Fig. 4, are similar to those of Section 4.2.2.

B Train time vs. #Parameters

Figure 5 shows the training time and number of parameters of each of our novel and baseline models. We make note here that our model training time is on the order of 12 hours each even though our model size is significantly smaller than most canonical LlaMa models to be able to fit on a single GPU.



(a) WikiQA Bleu–1 Score.

(c) WikiQA FuzzyWuzzy Score.

(b) WikiQA Bleu–2 Score.

(d) WikiQA Rouge–L Score.

Figure 4: Out-of-Distribution evaluation on the WikiQA dataset (general question answering). Novel models consistently have lower inference time compared to deeper baselines with no significant decrease in evaluation metrics. A final verdict, however, requires more rigorous testing in terms of larger models, more training data and longer training times.

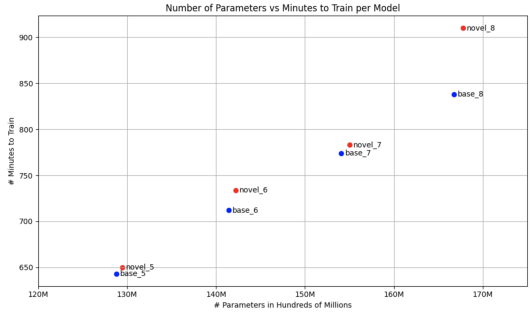


Figure 5: Number of parameters vs model training time for each model.