Not So Speculative Decoding

Zack Ankner **Rishab Parthasarathy** **Chris Rinard**

Abstract

As Large Language Models (LLMs) are increasingly adopted for real-world applications, the costs associated with these models are largely shifting from pretraining to inference. Recent approaches to speeding up LLM inference have proposed the parallel decoding framework, where a shallow model conditioned on the base LLM's representation of the sequence to predict multiple tokens into the future. However, with current methods, the prediction of the i^{th} future token is not conditioned on the previously predicted future tokens, i.e. $(i-1)^{th}$, $(i-2)^{th}$, ... future tokens. We propose to make future token prediction heads sampling aware by conditioning their predictions on both the most recent hidden state of the LLM and the input embeddings of the tokens that have already been generated. On MT-Bench, a real world inference task, our method leads to a 2.33x improvement over standard LLM inference throughput and a 1.07x improvement over the previous state-of-the-art parallel decoding framework's inference throughput.

1 Introduction

Recently, Large Language Models (LLMs) like ChatGPT (OpenAI, 2022) have revolutionized the way people interact with computers. While vast efforts have gone into the pre-training of autoregressive LLMs, users only interact with the resulting model during inference, i.e. when the model is predicting completions to prompts. Even though parallel prediction of the sequence is performed during pre-training and sequential decoding is performed at inference time, traditionally the same model architecture is used across both settings. As this model architecture is optimized for pre-training, inference is typically slower and less efficient. Prior to the development of modern LLMs, this tradeoff was considered beneficial, as LLMs were not widely used, rendering the amount of inference performed much lower than the amount of compute devoted to pre-training. However, with the widespread adoption of LLMs, inference costs will rise significantly. Hence, to mitigate economic and compute costs of these overall systems, we must now focus on the task of maximizing the efficiency of inference.

To formalize the difference in pre-training and inference, pre-training is a *compute-bound* problem since each layer in the LLM needs to process every single token in the sequence for each sequence in the batch in parallel, i.e. there is lots of work to be done by a model layer each time its loaded. On the other hand, inference, especially in the low batch size setting, is *memory-bound* since only batch-size number of tokens are processed at a time so the biggest bottleneck is the time taken to transfer *large* weight matrices from GPU memory to the GPU's compute units. This observation that LLM inference is frequently memory-bound raises the question of whether modifications to the original LLM architecture after pre-training can improve inference speed. Solving this issue could mitigate the significant compute usage and environmental impact of LLMs in the inference stage.

Recent work has proposed to improve LLM inference by using a smaller auxilary model to decode multiple tokens in parallel. These *speculated* tokens are then verified in parallel by the original larger model, and only tokens which agree with the larger model's predictions are sampled. This framework is based on the observation that since LLM decoding is memory-bound, we can increase the amount of computation we perform without decreasing speed. Thus, if we increase the compute we perform but decrease the number of times weights are read from memory, we will achieve a speedup. Speculative decoding (Leviathan et al., 2023) is an instance of parallel decoding, in which the auxiliary model is simply a significantly smaller Transformer model. This work focuses on a different parallel decoding paradigm, namely Medusa decoding (Stern et al., 2018; Cai et al., 2023), which attaches a collection of future token prediction heads that take the form of shallow multi-layer perceptrons (MLPs) to the LLMs hidden states. Intu-

itively, if an LLM's hidden representation encodes information to predict the next token the information it contains should also be sufficient to predict the next-next token, next-next-next token, and so on.

While Medusa decoding provides a significant speedup, we make the observation that the Medusa head predictions are not conditioned on previous Medusa head predictions. Essentially, when we are predicting i tokens into the future, the prediction is not aware of the $(i-1)^{th}, (i-2)^{th}, ...$ predictions. This leads to a significant degradation in future token prediction quality that increases as we decode further into the future. To address this, we present a generalizable extension of speculative decoding, called not so speculative decoding, that further improves the approximation quality of Medusa heads by conditioning their predictions on the tokens which have already been sampled. We implement this sampling aware conditioning by setting the inputs to the Medusa head to be the base LLMs representation of the sequence so far, as well as the input embeddings of the tokens which had been sampled.

Overall, we improve the quality of the parallel decoding framework, enabling us to decode further into the future for a single pass of the base LLM. We demonstrate that this leads to improved decoding efficiency, setting a new state of the art for parallel decoding throughput.

2 Related Work

Parallel decoding. The first work to leverage speculative decoding proposed to attach additional future token prediction heads, using MLPs, to the final hidden state. Thus, the model could predict multiple tokens cheaply by running the heads after each forward pass of the model, instead of requiring one forward pass for each token generated (Stern et al., 2018). This batch of future predicted tokens, generated by the heads, is then scored in parallel by the base LLM; if the score assigned is above a predetermined acceptance threshold, the batch of tokens is accepted; otherwise, the tokens are re-sampled and the process is repeated.

Other works have built on this by implementing paradigms where instead of querying the base model, an approximation model is trained and queried instead (Leviathan et al., 2023). However, these models suffer from low token acceptance rates, as it is difficult to train approximation models that estimate the outputs of a larger model well. Hence, these frameworks add complexity without significantly improving the throughput of these models, because they condition on input embeddings rather than the final hidden states of a transformer, leading to limited expressivity.

Medusa decoding. Medusa (Cai et al., 2023) makes the observation that the future token prediction heads used in Stern et al. (2018) are a poor approximation to the actual future tokens the base LLM would output if run sequentially. Medusa fixes this problem by generating a large set of possible completions, instead of generating a single possible completion, and then scoring each of the possible completions using the base model and selecting the best scoring sequence. While their decoding strategy improves the quality of future tokens, it further increases the computational cost as more completions need to be scored. Also, since each head is independent, their model does not encode the conditional dependence between tokens, an issue our framework directly will address.

Flash Decoding Outside of the field of speculative decoding, other inference research has instead focused on improving kernels for decoding. Such kernels may split up attention between multiple machines to improve matrix multiplication core utilization, or approximate the denominator of Softmaxes for online calculation of attention values (Dao et al., 2023) (Hong et al., 2023).

These approaches have significantly reduced the memory load of attention, but do not solve the underlying issue with inference, i.e. the sequential nature of the model. As such, our solution can be used together with Flash Decoding approaches to speed up base model calls, making these two methods mutually beneficial.

Other Inference Speedups While we focus on decoding-based speedups, the majority of inference research today centers on parallelism and deployment efficiency. One example of such a framework is Orca, which improves the efficiency of inference by scheduling each forward pass of the model, instead of scheduling per batch. In doing so, the model utilization increases significantly, but the base model performance does not change at all (Yu et al., 2022).

Alternatively, work like vLLM instead implements a new framework for attention, using ideas from virtualization in hardware. Using these tech-

niques together with a paged key-value cache reduces the number of computations during inference, increasing speed and reducing the memory overhead. However, vLLM still does not solve the basic architectural issue that each token generated requires the full compute of a forward pass (Kwon et al., 2023). Hence, just like Flash Decoding, these alternative inference speedups go hand-in hand with grounded multi-token decoding, solving systems and hardware level challenges, while our framework attacks model-level issues.

3 Methods

In this section we describe the Medusa Decoding framework (Cai et al., 2023) and describe our architectural modification to Medusa Heads.

3.1 Background Models

In traditional decoding, a model must be queried m times to produce m tokens. However, Stern et al. (2018) proposed a parallel decoding scheme by attaching multiple small multilayer perception (MLP) heads which, as a function of the base model's representation of the data seen thus far, predict future tokens greater than one timestep away. Cai et al. (2023) made the observation that the quality of future tokens the heads predict is poor when greedily selecting only the most likely completion. Thus, they propose to instead look at the top-k next token predictions from each head and select the best completion as scored by the base model, a technique they name Medusa. We formally describe the Medusa framework below.

Notation. Let V be our vocab and $x_{\leq t}$ be our input sequence up till time t. Our base transformer parameterizes a next token distribution, namely $p(x_{t+1}|x_{\leq t})$.

Medusa heads. We follow Cai et al. (2023) and refer to the lightweight future token heads as *medusa heads*. Formally, let $p_1 = p$, the base transformer next token distribution. The Medusa heads are a set of lightweight auxiliary models $p_2, ..., p_k$ where $p_i(x_{t+i}|x_{\leq t})$ is the probability that the i^{th} future token will be x_{t+i} given the input sequence up to time t. While this definition does not imply any constraints on the type of architecture for Medusa heads, in practice each head is a small multilayer perceptron (MLP) conditioned on the base transformers's representation of the input sequence $x_{\leq t}$. The base transformers representation of $x_{\leq t}$ is taken to be the transformers final hidden state for token x_t.

Tree decoding. While Medusa heads specify a distribution over future tokens, we still need a way to sample from that distribution. The simplest way to sample from the distribution is to sample greedily from each Medusa head, meaning our prediction for the i^{th} future token is $\hat{x}_{t+i} = \text{argmax}^1_{x_{t+i}} p_i(x_{t+i}|x_{\leq t})$. However, the simple greedy sampling strategy often predicts the wrong future token as the top-1 Medusa head predictions do not align with the base transformers predictions. To overcome this, Cai et al. (2023) propose to instead look at the top-k predictions for each head, and see if any of those predictions match the base transformers prediction. The set of all possible completions, which we refer to as the *tree candidates*, is the Cartesian product of the top-k predictions for each Medusa head.

Tree verification. Given the tree candidates, which represent all possible completions, we need to score each candidate according to the base model. We use the verification criteria that a given token in a tree candidate is only accepted if it matches the token the model would have produced had it performed greedy decoding. To compute the base transformer's completions for each candidate, we extend our original sequence with each tree candidate. Then, we apply an attention mask such that each candidate completion only attends to its own tokens and the original sequence, not the other candidates. That is to say, building our tree candidates as the product of all Medusa heads induces a tree-structure dependence on future token predictions, and we define the attention mask to encode this dependence, as depicted in the first diagram in Figure 1.

Finally, to sample the completion of $x_{\leq t}$, for each candidate we truncate it to the number of tokens which agree with the tokens produced using greedy sampling from the base transformer. We then select the best truncated candidate (the maximum length of tokens matching the base model) as our completion.

3.2 Improving upon Medusa

Our work improves the performance of Medusa decoding by improving the fidelity of the Medusa heads, specifically by making Medusa heads sampling-aware. The key problem with Medusa

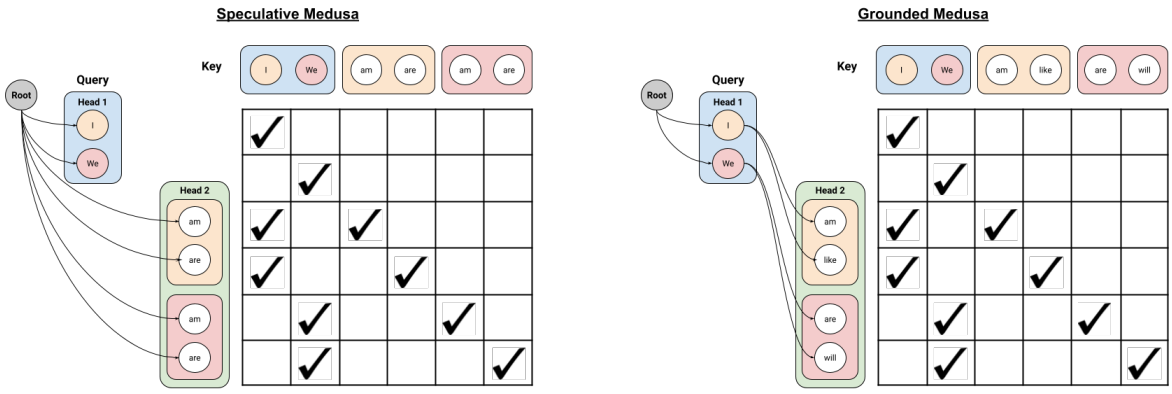

Figure 1: A depiction of speculative vs grounded Medusa tree attention. Speculative Medusa has the same head outputs regardless of previously speculated tokens, while ours has a conditional dependence that allows grounded Medusa to explore more of the state space. Note that the tokens generated by grounded Medusa in this example match the tense of the previous candidates, where speculative Medusa does not.

heads is that while the i^{th} Medusa head parameterizes the distribution for the i^{th} future token, its parameterization $p_i(x_{t+i}|x_{\leq t})$ is invariant to the tokens predicted by earlier Medusa heads as the distribution is not conditioned on $\hat{x}_{t\leq j\leq t+i-1}$. This leads to poor performance, as later Medusa heads are not aware of the tokens sampled by previous Medusa heads.

Grounded Medusa heads. Our modified Medusa heads, which we name *grounded Medusa heads*, are instead designed to be sampling aware as they are conditioned on the previously speculated tokens. Namely, the grounded Medusa heads parameterize the distribution $p_i(x_{t+i}|x_{\leq t}, \hat{x}_{t<j<t+i})$. While there are multiple ways to implement this conditioning, unless otherwise specified we ground the Medusa head MLP by setting its input to be the base transformer's representation of $x_{\leq t}$ concatenated with the base transformer's input embeddings of the tokens sampled thus far. Formally, the input representation is

$$f(x_{\leq t}) \oplus E_{\hat{x}_{t+1}} \oplus ... \oplus E_{\hat{x}_{i-1}}$$

where $\oplus$ denotes vector concatenation, $f(x_{\leq t})$ is the base transformer's representation of the sequence, and E_x is the transformer's input embedding for token x.

Grounded tree candidates. As the original Medusa heads did not have any dependence on earlier Medusa heads, the tree candidates could simply be constructed by running each Medusa head independently, then taking the Cartesian product of each heads top-k predictions. Since grounded Medusa heads introduce a dependence between heads, this is no longer possible. Instead, while we still generate the same number of tree candidates to verify with the exact same tree structure, we do so in a sequential process that respects the tree dependence, as seen in the second diagram of 1.

After making the above changes to the Medusa framework, all other steps such as tree masking and verification are left unchanged.

4 Evaluation

In this section we describe the setup and methodology for training and evaluating our proposed grounded Medusa heads, as well as discussing the results.

4.1 Methodology

Models For all experiments, we use Vicuna 7B (Vicuna, 2023) as the base transformer. It is a 7 billion-parameter decoder-only transformer that was fine-tuned from Meta's LLaMA-7B on multi-turn conversation data. For all Medusa heads, whether grounded or not, we use an MLP with one hidden layer of dimensionality `4096`, and attach four Medusa heads to the base transformer.

Training We freeze the parameters of the base transformer and only fine-tune the added Medusa heads. We fine-tune the Medusa heads on the ShareGPT dataset (Vicuna, 2023), a collection of 121 thousand multi turn conversations. We fine-tune using the `Adam` optimizer (Kingma and Ba, 2017) with $\beta_1 = 0.9, \beta_2 = 0.99$, a cosine-decay

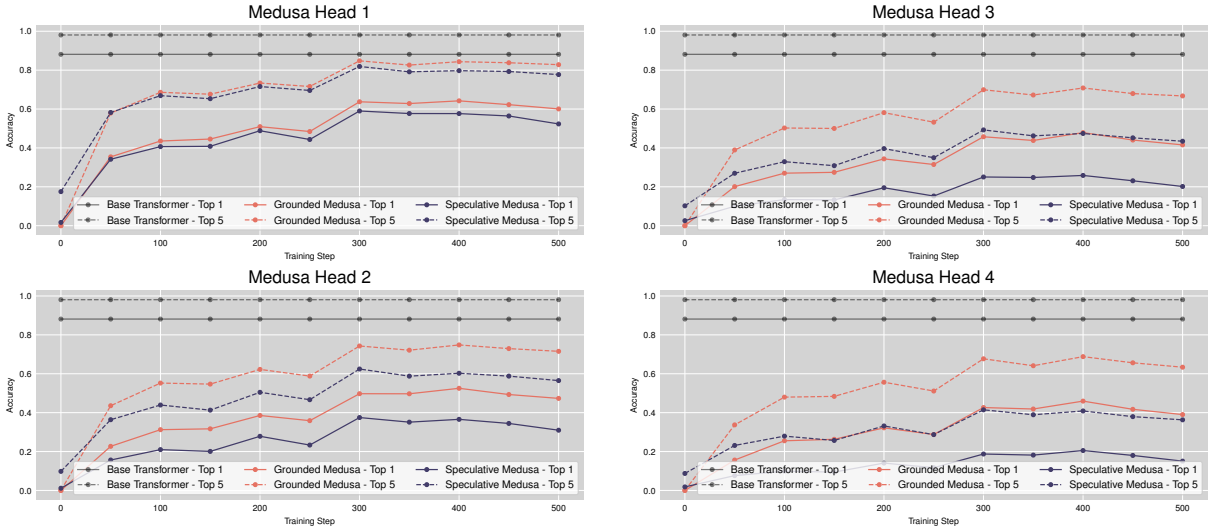

Figure 2: Top-1 and Top-5 token prediction accuracies throughout fine-tuning for grounded and speculative Medusa heads. We also include the token prediction accuracy of the base transformer.

learning-rate schedule with warmup that has an initial and final learning-rate of `1e-4` and `0`, respectively, a training duration of one epoch, a sequence-length of 2048, and a batch-size of 128. We conduct all training on `8x Nvidia A100s` using the HuggingFace trainer (Gugger et al., 2022) and the Medusa codebase (Cai et al., 2023).

Downstream Evaluation To evaluate the downstream performance of the Medusa models, we use the MT-Bench dataset (Zheng et al., 2023) of multi-turn questions and answers. These questions are collected from a variety of subjects, such as humanities, science, and math, and are asked to the model as a chain of inputs. On MT-Bench, we measure the compression ratio of our framework (the average number of future tokens accepted per decoding step) and the throughput of the models. These values are computed by running experiments that keep track of the time each generation takes, along with the number of calls to the model's forward function.

5 Results

5.1 Medusa Head Accuracy

To test whether grounding Medusa heads provides higher quality token predictions, we test the accuracy of the Medusa heads on the ShareGPT dataset throughout training. While ShareGPT is the dataset on which the models are trained, we perform training in the online setting (only one epoch), so the training accuracy is equivalent to the model's performance on a hold-out validation split. We plot the top-1 and top-5 accuracy of the grounded and baseline Medusa heads in Figure 2. We additionally plot the original model's token prediction accuracy for reference. As seen in Figure 2, grounding Medusa heads leads to better top-1 and top-5 token prediction for all Medusa head positions examined. Furthermore, the accuracy improvement of grounded Medusa heads over speculative Medusa heads grows as we predict tokens further into the future. While grounded and speculative heads have similar performance for the first Medusa head, by the fourth Medusa head the top-1 accuracy of the grounded Medusa head outperforms the top-5 accuracy of the speculative Medusa head. These results demonstrate that grounding Medusa heads allows for higher-quality predictions of future tokens.

5.2 MT-Bench Evaluation

To determine whether grounding Medusa heads leads to improved decoding speed on downstream tasks, we evaluate all model variants on MT-Bench. For each model we compute the *compression rate*, defined as the number of accepted tokens per parallel decoding step, and the *throughput*, defined as the number of tokens generated per second. We present the results in Figure 3 and Figure 4 respectively. We find that grounded Medusa heads achieve the best token throughput of any method examined on all tasks in MT-Bench Specifically, compared to the throughput of the baseline and speculative Medusa heads, grounded Medusa heads are an overall 2.33x and 1.07x improvement respectively. The reason for grounded Medusa heads

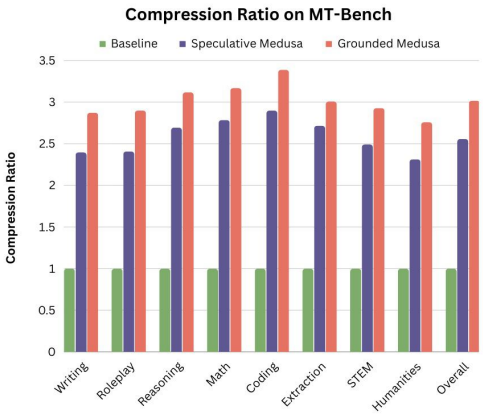

Figure 3: Number of tokens accepted per base LLM forward pass for each task category in MT-bench.

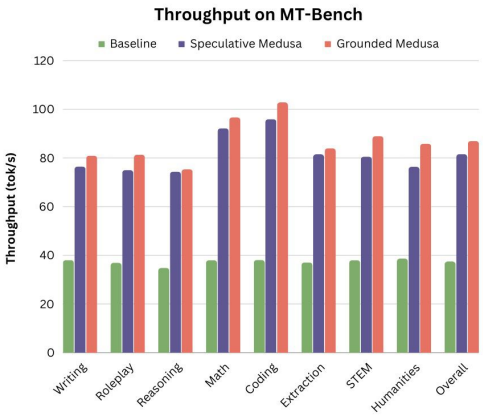

Figure 4: Decoding throughput for each task category in MT-bench.

throughput improvement over the baseline is its better compression. On average, grounded Medusa heads can predict 0.43 tokens further into the future than speculative Medusa heads. These results demonstrate that the improved modeling quality of grounded Medusa heads leads to improvements in generation speeds on downstream tasks.

5.3 Ablations

To further understand the success of grounded Medusa heads, and to determine if there are more preferment grounded Medusa head setups, we ablate the grounded Medusa head design.

5.3.1 MLP Depth

The first intuitive change we seek to explore is whether increasing the number of layers in the head MLP, which increases capacity, would significantly impact performance. We investigate whether increasing the capacity of the grounded Medusa

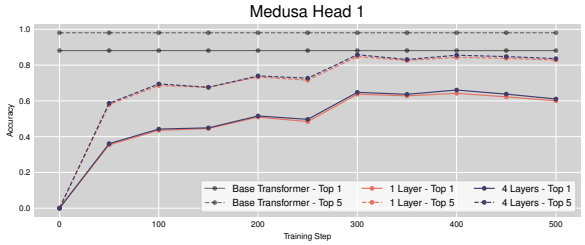

Figure 5: Performance of different MLP depths in our model.

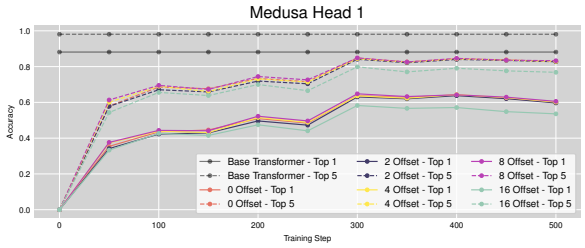

Figure 6: Performance of different hidden state offsets.

heads would improve their prediction quality We test this by using a deeper MLP for our grounded Medusa head, specifically we increase the number of hidden layers from 1 to 4. We present the results in Figure 5.

Overall, we find that the deeper MLP confers a neglibible improvement in head accuracy. This may be attributed to the fact that the hidden state of a transformer is sufficiently expressive that a one-layer MLP is enough for predicting multiple tokens ahead.

5.3.2 Hidden State Offset

Medusa heads are only made aware of the history of the sequence $x_{\leq t}$ through the base transformer's final hidden representation of x_t. As previous work has suggested that the mutual information between the input and the transformer's hidden representation rapdily decreases with the depth of the hidden representation (Voita et al., 2019), we investigated whether representing $x_{\leq t}$ with an earlier hidden layer's representation would more faithfully encode the sequence and thus lead to higher Medusa head prediction quality. To test this, we evaluate hidden state offsets of 0, 2, 4, 8, and 16, in Figure 6

We find that for offsets of up to 8, there is a negligible change in head accuracy, while an offset of 16 (or half the model depth in the LLaMA-7B model we use) significantly degrades performance. This leads us to the conclusion that expressivity does not change drastically for the multi-token prediction task over the last hidden states of an LLM, as the

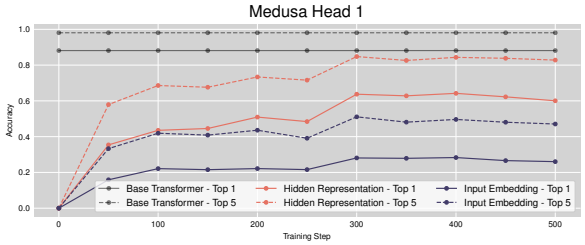

Figure 7: Performance of input embeddings against final hidden state.

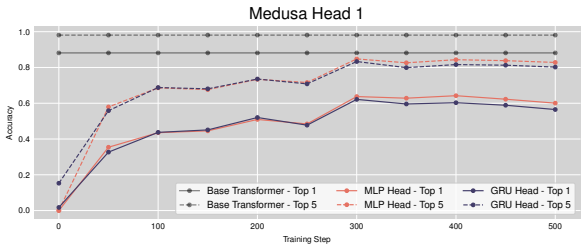

Figure 8: Performance of GRU heads.

overspecialized final layer still provides optimal performance on the task.

5.3.3 Input Embeddings

To further verify the expressivity of the final hidden state, we also compare performance against using input embeddings. Since other parallel decoding frameworks, namely speculative decoding (Leviathan et al., 2023), operate on the input tokens themselves which are then passed to a lightweight transformer, we wish to test whether such a framework contains sufficient contextual information compared to the output of the larger transformer. We evaluate the head quality conditioned on only input embeddings in Figure 7.

Overall, we find that using input embeddings vastly under performs the final hidden state of the transformer, showing the advantage of using a deep representation.

5.3.4 Reccurent Medusa Heads

So far, we have only investigated Medusa heads that use an MLP architecture. While these MLPs are capable of representing the sequential dependence of text, we investigate whether using an architecture with a sequential inductive bias, namely a GRU (Cho et al., 2014), would improve Medusa head quality. The initial hidden state of the GRU is the transformer's hidden representation of the sequence, and the input at each time step is the input embedding of the token sampled the step before.

We present the results in Figure 8. We observe that the RNN heads do not change performance significantly compared to the single layer MLP. This may be because the GRU does not encode enough contextual information, meaning that its expressivity does not improve enough over the final hidden state of the LLM itself.

6 Discussion

Overall, we observe that conditioning Medusa heads on the outputs of previous Medusa heads produces improvements across the board. It leads to an increase in Medusa head accuracy which leads to an improvement in inference speed. This is likely because knowing the outputs of the previous heads allows later heads greater expressivity–instead of having to produce the most likely results without any contextual information, they can specialize their outputs to what they have already seen.

We also ablate the key components of the Medusa head. We find that most interventions do not have a significant impact on the performance of grounded Medusa heads. This is not to say that the current grounded Medusa head framing is optimal. Future work will include investigation into other design choices, such as using a self-attention layer (Vaswani et al., 2017) as the Medusa head architecture. There is also work to be done in exploring the sampling scheme such as using beam search like methods to generate our candidate tree rather than a fixed candidate tree.

7 Conclusion

In this paper, we present a method for decoding multiple tokens with each forward pass of a transformer model to lighten the significant burden of LLM inference. We build on the work of the Medusa framework (Cai et al., 2023), which introduces independent heads that use the final hidden state of the transformer model to predict a number of tokens into the future. We propose to use heads conditioned on the predictions of previous heads rather than independent predictors to increase the expressivity of Medusa heads. In doing so, we observe that our method results in more accurate heads, and thus a greater number of correctly sampled tokens per forward pass of the model. This leads to a significant lift in throughput for grounded Medusa heads as compared to speculative Medusa heads. Hence, our method could greatly reduce the computational burden of LLM inference.

Impact

Our work is likely to increase the usability and deploy-ability of Large Language Models (LLMs) at scale by making inference more efficient. This means that our method can accelerate progress in any application where LLMs are used, as our method is agnostic to purpose of the underlying LLM. Hence, grounded Medusa heads can be used in use cases from creative writing to developing code in industry.

In the context of our framework, increasing the speed of inference has a number of positive contributions, as reducing the compute requirements of LLM inference reduces the amount of emissions produced by compute centers. On top of reducing emissions, reducing decoding speed can unlock new LLM applications, like LLM agents, which require the generation of lots of text, and thus are bottle-necked by decoding speed. In improving decoding speeds, LLM agents can be deployed in more environments, improving the accessibility of the internet through online assistants, to LLMs that can interact with the real world through robotics. However, while there are many use cases of LLMs which contribute to society in a positive manner, there are also many dangerous LLM use cases. Especially given that our framework does not control the base behavior of the LLM, only changes the inference speed, we must be careful about potential negative applications. For example, more rapid inference could lead to increased production of misleading fake news and misinformation in our current political environment. Also, unlocking more complex applications of LLMs like agent-based models could also exacerbate societal divides due to the unequal access to LLM-based technologies.

To these ends, we encourage work along multiple axes:

1. First, exploring how to **control generation capabilities**, as this can directly confront issues of systematic bias and misinformation in LLM responses.

2. Second, exploring methods for increasing equitable access to LLM technology, ensuring that these technological advancements reach a wide range of people from different socioeconomic backgrounds.

3. Finally, are there ways to integrate this technique with other inference speedups, making it more feasible for a consumer to host an LLM without relying on big corporations?

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. 2023. Medusa: Simple framework for accelerating llm generation with multiple decoding heads. https://github.com/FasterDecoding/Medusa.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-decoding for long-context inference. https://crfm.stanford.edu/2023/10/12/flashdecoding.html. Accessed: 2023-11-28.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. 2023. Flashdecoding++: Faster large language model inference on gpus.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

OpenAI. 2022. Introducing chatgpt.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. volume 31.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Team Vicuna. 2023. A framework for learning predictive structures from multiple tasks and unlabeled data.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA. USENIX Association.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.