Draft on the Fly: Adaptive Self-Speculative Decoding using Cosine Similarity

Michael R. Metel[1], Peng Lu[2], Boxing Chen[1], Mehdi Rezagholizadeh[1], Ivan Kobyzev[1]
[1]Huawei Noah's Ark Lab, [2]DIRO, Université de Montréal, Canada
Correspondence: michael.metel@h-partners.com

Abstract

We present a simple on the fly method for faster inference of large language models. Unlike other (self-)speculative decoding techniques, our method does not require fine-tuning or black-box optimization to generate a fixed draft model, relying instead on simple rules to generate varying draft models adapted to the input context. We show empirically that our lightweight algorithm is competitive with the current SOTA for self-speculative decoding, while being a truly plug-and-play method.

1 Introduction

The main bottlenecks of transformer-based large language model (LLM) inference are the sequential generation of tokens, due to its autoregressive nature, and the need to reload the KV cache, causing inference to be memory bandwidth bound (Shazeer, 2019). Both of these bottlenecks prevent the full use of computing resources. Speculative decoding is a method to decrease the inference time of LLMs by taking advantage of this available compute capacity through the observation that the time of processing a single input token is roughly the same as processing a small number of tokens in parallel.

After processing the input context and generating the first token t_1 using a model M, instead of sequentially generating tokens until completion, speculative decoding uses a smaller *draft* model D to generate a short sequence of tokens $[\hat{t}_2, \hat{t}_3, ...]$. The tokens $[t_1, \hat{t}_2, \hat{t}_3, ...]$ are then processed in parallel by M for verification. The first rejected token $\hat{t}_r$ is replaced by M's predicted token t_r. The process is then repeated, generating a new sequence of tokens $[\hat{t}_{r+1}, \hat{t}_{r+2}, ...]$ for verification.

When using greedy decoding, the verification process is to simply ensure that $\hat{t}_i = t_i$ for each drafted token $\hat{t}_i$. When sampling is used, a clever method has been devised to accept tokens which ensures that the distribution they were sampled from matches that of M (Chen et al., 2023; Leviathan et al., 2023). Simply put, speculative decoding has the ability to speed up inference without sacrificing accuracy.

The ability to speed up inference also stems from the varying difficulty in predicting the next token over the generated sequence, where even quite small draft models are capable of predicting a percentage of tokens correctly. Choosing a D to maximize speedup then requires balancing its speed of token generation with the accuracy of the tokens it generates.

In this work we propose a self-speculative decoding algorithm, where D is chosen as a subnetwork of M, generated on the fly and adapted to the input context. Compared to the current state of speculative decoding described in Section 2, our method is simple to implement as a truly plug-and-play method, avoiding the challenges of finding a fixed draft model using fine-tuning or black-box optimization. In Section 4 we observe that despite the simplicity of our proposed Adaptive Draft Model Generator (ADMG, Algorithm 1), our method is competitive with the current SOTA of self-speculative decoding.

2 Related Work

Speculative decoding was first developed in (Chen et al., 2023; Leviathan et al., 2023). These works proposed using a second smaller model to draft tokens from. Acquiring such a draft model is in general challenging, and will typically require fine-tuning (Cai et al., 2024; Li et al., 2024; Zhang et al., 2024). Sufficient memory must also be available to store a second model as well as save both models' KV caches.

To alleviate the need for a separate draft model, self-speculative decoding was developed in Draft & Verify (Zhang et al., 2024), where D is chosen as a subnetwork of M by removing attention and

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 2267–2272
November 12-16, 2024 ©2024 Association for Computational Linguistics

MLP layers. Choosing which layers to remove to minimize the average inference time is a binary optimization problem (NP-hard) of a black-box objective function. An approximate solution to this problem is found using Bayesian optimization with a small number of training samples. This is a time consuming process (see Section 4), with the solution depending on the model architecture, dataset, and computing environment.

Kangaroo (Liu et al., 2024) can be seen as a hybrid approach between using a fine-tuned draft model and self-speculative decoding, where D consists of the first $l \in \mathbb{N}$ layers of M, a fine-tuned adapter layer, and the LM Head of M.

Medusa (Cai et al., 2024) is an adjacent method to speculative decoding which uses multiple decoding heads to predict tokens in parallel, avoiding the challenges of obtaining an appropriate draft model. In its simplest form, these additional heads must be trained with the original weights of M kept frozen.

EAGLE (Li et al., 2024) also does not use a separate draft model, but drafts tokens using an additional decoder layer which must be trained. The added decoder layer takes as input the embedding of the last generated token and the input to the LM head of the penultimate generated token, with its output fed into the LM head of M to predict the next token.

Compared to these related works, a main benefit of our method is that it does not require the use of any fine-tuning or black-box optimization. Fine-tuning can be challenging when the training set of M is not available, as this can result in a shift in the output distribution of D compared to M, making the token verification stage of speculative decoding more challenging (Cai et al., 2024). Similarly, for Draft & Verify, the chosen draft model can be sensitive to the calibration data used, and its performance may suffer if an appropriate dataset is not available, see Section 4.1 for further discussion.

3 Adaptive Self-Speculative Decoding

In order to generate draft models on the fly, our main technique is to use a simple thresholding rule to remove layers from the original model based on the cosine similarities of the hidden states of the input context.

	Attention	MLP
Minimum ACS	0.581	0.854
Maximum ACS	0.998	0.991
Mean ACS	0.953	0.942
Median ACS	0.977	0.960
Mean Time (ms)	1.339	0.768

Table 1: Statistics of the average cosine similarity (ACS) and the average compute time of the layers of Llama-2-13B based on the input context of 1000 training set samples of CNN/DM.

3.1 Removing Attention Layers Based on Cosine Similarity

Our proposed Adaptive Self-Speculative Decoding (ASD) method uses the cosine similarity of the hidden states before and after each attention layer as an estimate of its importance, based on the intuition that the closer the cosine similarity is to 1 for a given attention layer, the smaller the effect of removing it should be on the model's accuracy. Our decision to only remove attention layers is motivated by observations from computing the average cosine similarity (ACS) of every attention and MLP layer, and the average compute time of attention and MLP layers of Llama-2-13B over the input context of 1000 samples of the training set of CNN/DM, summarized by the statistics given in Table 1. The following three observations motivated the focus on only removing attention layers based on cosine similarity:

1. The higher mean and median ACS of attention layers indicate that on average more attention layers than MLP layers can be removed.

2. The range of the ACS is larger for attention layers, considering Maximum ACS - Minimum ACS, allowing for an easier differentiation of removable attention layers.

3. The average attention compute time is 1.74× greater than that of MLP layers.

Considering the added compute time to inference from having to calculate the cosine similarity and determine which layers to remove, we concluded that focusing solely on attention layers would give the best return in terms of inference speedup, given that more attention layers will be removable, it will be easier to determine which layers to remove, and a greater reduction in draft time will be achieved per removed layer.

3.2 Implementation & Further Heuristics

When performing inference, the input context serves as a natural calibration set to adaptively choose which attention layers to remove from M to form D for the given instance.

The first step of inference is to pass the entire input context through the original model M in parallel to generate the first token and populate the KV cache. As the hidden states of the input pass through each layer, ASD computes the ACS of each attention layer over the input sequence length in order to choose D for the following token generation stage.

In particular, let $X^l \in \mathbb{R}^{S \times H}$ and $Y^l \in \mathbb{R}^{S \times H}$ be the hidden states before and after the l^{th} attention layer for $l = 1, ..., L$, where S is the input sequence length, H is the hidden size, L is the number of layers, and assuming a batch size of 1. We first compute the ACS over the input sequence, $C_l := \frac{1}{S} \sum_{s=1}^{S} \frac{\langle X_s^l, Y_s^l \rangle}{||X_s^l||_2 ||Y_s^l||_2}$. For a fixed constant $\alpha \in (0, 1)$, all attention layers with $C_l \geq \alpha$ are removed from M to generate D for the current generation task, which will be refered to as *CS thresholding*.

Even though this method enabled significant inference speedup on its own, in order to further speed up inference, we found two simple deterministic rules to be effective:

1. Remove every m^{th} attention and MLP layer.
2. Do not remove the last n attention and MLP layers when using rule 1 or CS thresholding.

An insightful hypothesis for the effectiveness of rule 1 was proposed in (Sajjad et al., 2023): In an M with redundancy in its layers, neighbouring layers contain similar information, hence if layer l is removed, its information is still largely contained in layers $l - 1$ and $l + 1$.

Rule 2 was initially proposed to preserve the first and last n layers, but by choosing $n < m$ in our implementation (Table 2), and from low observed ACS in the first n layers, this extra condition became unnecessary. The analysis in (Sun et al., 2024) gives support for this rule, as it was found that middle layers, matching closely to layers $\{l : n < l \leq L - n\}$ for our given n in Table 2, share the same representation space, allowing the output from layer k to be interpretable by layer l, for $l > k$. We also note that the importance of the last few layers has been previously observed in (Gromov et al., 2024).

Our Adaptive Draft Model Generator (ADMG) is presented as Algorithm 1, which outputs the attention and MLP layers to remove. An ablation study on ADMG is presented in Section 4.1.

Algorithm 1 Adaptive Draft Model Generator (ADMG)

```
Input: $C \in [-1, 1]^L$; $\alpha \in (0, 1)$; $m, n \in \mathbb{N}$
remove$_{\text{ATTN}} = \{l : C_l \geq \alpha, l \leq L - n\}$
remove$_{\text{MLP}} = \{l : l = jm, m \in \mathbb{N}, l \leq L - n\}$
remove$_{\text{ATTN}} =$ remove$_{\text{ATTN}} \cup$ remove$_{\text{MLP}}$
Output: remove$_{\text{ATTN}}$, remove$_{\text{MLP}}$
```

4 Experiments

We compare our method to the self-speculative decoding algorithm of Draft & Verify, which is most similar to our work. We conducted the same (fine-tuned) Llama-2-13B experiments found in the body of their paper without changing any of the chosen hyperparameters.

The experiments cover three models and three datasets: Llama-2-13B and Llama-2-13B-Chat (Touvron et al., 2023) evaluated on 1000 random test set samples of CNN/DM (Nallapati et al., 2016) and XSUM (Narayan et al., 2018), and CodeLlama-13B (Rozière et al., 2023) evaluated on HumanEval (Chen et al., 2021). CNN/DM and XSUM are both summarization tasks whereas HumanEval is a Python code generation task. The test sets were sampled using the same random seed as in Draft & Verify's experiments.

An important factor which determines the level of inference speedup is the number of tokens drafted before verifying their accuracy with M. We used the Adaptive Draft-Exiting Mechanism proposed in Draft & Verify (Zhang et al., 2024, Section 3.4), keeping the hyperparameters equal to their tuned values (Zhang et al., 2024, Table 6) to match their experiments.

We conducted all experiments using two V100-32GB GPUs. Draft & Verify relies on Bayesian optimization to find their draft model, which is sensitive to the computing environment. In order to get the best performance from their method, we reran their optimization code to generate draft models tuned to our environment. In total three draft models were generated, one for each model, which took on average 11 hours to generate per model.

For our ADMG (Algorithm 1), we used the hyperparameters in Table 2. The only difference

Model	α	m	n
Llama-2-13B	0.985	3	2
Llama-2-13B-Chat	0.985	3	2
CodeLlama-13B	0.985	4	3

Table 2: Hyperparameters used for ADMG (Alg. 1)

Model	T	Method	CNN/DM	XSum
Llama-2 -13B	0.0	D&V	**1.495**×	**1.415**×
		ASD	1.443×	1.383×
	0.2	D&V	**1.479**×	**1.383**×
		ASD	1.422×	1.350×
Llama-2 -13B-Chat	0.0	D&V	1.238×	**1.143**×
		ASD	**1.247**×	1.110×
	0.2	D&V	1.219×	**1.134**×
		ASD	**1.238**×	1.102×

Table 3: Inference speedup of Draft & Verify (D&V) and Adaptive Self-Speculative Decoding (ASD) methods compared to vanilla autoregressive generation using different temperatures (T).

over the experiments was that higher speedup was observed by incrementing m and n by 1 for the CodeLlama-13B experiments. Since CodeLlama-13B is fine-tuned for a more specific task compared to the other models, we believe that there may be less redundant layers which can be easily removed using rules 1 and 2.

The results of the experiments are presented in Tables 3 and 4. We observe that both methods have similar performance overall, with Draft & Verify performing better on the Llama-2-13B experiments, ASD having better performance on the CodeLlama-13B experiments, with mixed performance observed for Llama-2-13B-Chat.

4.1 Discussion & Ablation Study

It may be surprising that choosing a draft model by simple rules can give on par performance with a costly optimization method, but there are no guarantees on the solution quality when using normal Bayesian optimization to solve a discrete optimization problem by rounding the suggested continuous points, as was done in Draft & Verify. This may result in the algorithm getting stuck by revisiting previously sampled points (Luong et al., 2019), which was in fact observed when running their implementation.

Model	T	Method	Speedup
CodeLlama-13B	0.0	D&V	1.282×
		ASD	**1.365**×
	0.6	D&V	1.282×
		ASD	**1.340**×

Table 4: HumanEval inference speedup of Draft & Verify (D&V) and Adaptive Self-Speculative Decoding (ASD) compared to vanilla autoregressive generation using different temperatures (T).

Drafting Generation Rules	Speedup
CS thresholding	1.307×
CS thresholding & rule 2	1.340×
Rules 1 & 2	1.179×
ADMG	1.443×

Table 5: Ablation study on ADMG (Algorithm 1) showing the inference speedup using subsets of its 3 rules for Llama-2-13B on 1000 test samples of the CNN/DM dataset using greedy decoding.

The largest difference in inference speedup between Draft & Verify and ASD is observed in the CodeLlama-13B experiments. Given that HumanEval does not have a training set, Python samples from the StarCoder (Li et al., 2023) training dataset were used to generate Draft & Verify's D, as was done in their experiments. We believe the high relative speedup of ASD demonstrates the robustness of ADMG, and the potential sensitivity to differences in the distribution of the calibration and test datasets for their method. The implication of this being that in general, it may be challenging for Draft & Verify to remain on par with our method when it cannot be guaranteed that the difference between the calibration data and the data it processes through time will remain small.

We end this section by giving an ablation study on ADMG, showing the inference speedup achieved when generating draft models using subsets of its 3 rules, presented in Table 5. We observe that the majority of the inference speedup comes from CS thresholding, with increased inference speed when adding rule 2, and then again when adding rule 1 (ADMG). We also observe that even without using CS thresholding, inference speedup is achieved only using rules 1 and 2.

5 Conclusion

We have proposed a self-speculative decoding method to generate draft models on the fly which are adapted to the input context. Our method uses cosine similarity thresholding with simple layer removal rules to generate draft models without the need for any fine-tuning, black-box optimization, or training data. Despite our proposed method's simplicity, we found that it is competitive with the current self-speculative decoding SOTA, while being an easy to implement plug-and-play method.

Limitations

The inference speedup achieved by ASD relies on the existence of layers which when removed, do not significantly impact the model's accuracy. This reliance on the level of redundancy in the LLM's layers will limit the potential inference speedup of ASD.

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *ICML*, pages 5209–5235. PMLR.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling. *arXiv preprint arXiv:2302.01318*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The Unreasonable Ineffectiveness of the Deeper Layers. *arXiv preprint arXiv:2403.17887*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. In *ICML*, pages 19274–19286. PMLR.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. 2023. StarCoder: may the source be with you! *TMLR*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. In *ICML*, pages 28935–28948. PMLR.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. 2024. Kangaroo: Lossless Self-Speculative Decoding via Double Early Exiting. *arXiv preprint arXiv:2404.18911*.

Phuc Luong, Sunil Gupta, Dang Nguyen, Santu Rana, and Svetha Venkatesh. 2019. Bayesian Optimization with Discrete Variables. In *AI 2019*, pages 473–484. Springer.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *CoNLL*, pages 280–290. ACL.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *EMNLP*, pages 1797–1807. ACL.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950*.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Noam Shazeer. 2019. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv:1911.02150*.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. 2024. Transformer Layers as Painters. *arXiv preprint arXiv:2407.09298*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Scialom Thomas. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. Draft & Verify: Lossless Large Language Model Acceleration via Self-Speculative Decoding. In *ACL*. ACL.