Speculative Decoding with CTC-based Draft Model for LLM Inference Acceleration

Zhuofan Wen[1,4]**,Shangtong Gui**[1,2,4]**,Yang Feng**[1,3,4*]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
[2]State Key Lab of Processors,
Institute of Computing Technology, Chinese Academy of Sciences
[3]Key Laboratory of Al Safety, Chinese Academy of Sciences
[4]University of Chinese Academy of Sciences, Beijing, China
{wenzhuofan24z,guishangtong21s,fengyang}@ict.ac.cn

Abstract

Inference acceleration of large language models (LLMs) has been put forward in many application scenarios and speculative decoding has shown its advantage in addressing inference acceleration. Speculative decoding usually introduces a draft model to assist the base LLM where the draft model produces drafts and the base LLM verifies the draft for acceptance or rejection. In this framework, the final inference speed is decided by the decoding speed of the draft model and the acceptance rate of the draft provided by the draft model. Currently the widely used draft models usually generate draft tokens for the next several positions in a non-autoregressive way without considering the correlations between draft tokens. Therefore, it has a high decoding speed but an unsatisfactory acceptance rate. In this paper, we focus on how to improve the performance of the draft model and aim to accelerate inference via a high acceptance rate. To this end, we propose a CTC-based draft model which strengthens the correlations between draft tokens during the draft phase, thereby generating higher-quality draft candidate sequences. Experiment results show that compared to strong baselines, the proposed method can achieve a higher acceptance rate and hence a faster inference speed.

1 Introduction

Large Language Models (LLMs) have been applied to a wide range of text generation tasks such as machine translation and question answering due to their remarkable performance[1, 21, 6, 11]. In many applications, LLMs is required to produce a long generation to give explanations to its answer or perform chain of thought (CoT). Moreover, in some practical scenarios, LLMs have to resort to other applications to fulfill a task under the frame of AI agents where LLMs usually generate long outputs to communicate with other applications. All of these have a high demand for the inference speed of LLMs. However, most LLMs employ a token-by-token autoregressive generation paradigm, bringing on the severe inference delay problem, which obstacles the real applications of LLMs.

To mitigate the inference latency of LLMs, speculative decoding is proposed and proves to be a more efficient decoding strategy compared with autoregressive generation[12, 4]. Besides the base LLM, speculative decoding usually introduces a drafter model in the working flow that the draft model generates candidates for the next several tokens and the base LLM verifies the candidate and decides to accept or reject at some criterion. Once accept, then the winner candidate will be used as the output, otherwise, the base LLM will decode and generate the output. In this process, the draft

*Corresponding author: Yang Feng

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2923

model usually has few parameters and hence produces generations at a faster speed. Meanwhile, the LLM can verify draft tokens parallelly and takes less time than generating the next several tokens by itself. As only the candidate is accepted at a high rate, the decoding speed can be improved. It can be derived that the final inference speed is related to the decoding speed of the draft model and the acceptance rate of the candidate selected out by the draft model. Nevertheless, there is a trade-off for the draft model between its decoding speed and its performance related to acceptance rate.

Following the principle of speculative decoding, many works focus on optimizing the draft model to accelerate inference [15, 19, 5, 17] in which the methods based on non-autoregressive (NAR) generation have shown promising results. The NAR speculative decoding methods draft the next several tokens parallelly by predicting them independently based on the representation of the original LLM. Although these methods drafts at a high speed, they ignore the dependence between the next several tokens and sacrifice the performance as the cost. As a consequence, the speed of speculative decoding can be affected via the acceptance rate.

Based on these observations, we make efforts from the perspective of model performance by introducing dependency relationships during draft generation, aiming at achieving a higher acceptance rate. At this end, we propose a draft model based on Connectionist Temporal Classification(CTC) algorithm [9] which generates drafts in a non-autoregressive way with additional blank and repetitive tokens participating in. As during training the CTC-based draft model will count all the possible candidates sequentially that can generate the given ground truth when calculating the probability of the ground truth, the candidates with better dependency relationships will achieve higher probabilities. As a result, at inference the best candidate selected out by the CTC-based draft model will be more sequentially reasonable and hence can be accepted at a higher rate, ultimately leading to faster inference. For the rest of this paper, we refer to CTC-drafter as CTC-based draft model for short. Experiments on MT-bench show that the proposed method is able to draft sequences at a higher acceptance rate compared to strong baselines, thus achieving remarkable inference speedup.

Our contributions are as follows:

1. We introduce the CTC-based draft model to speculative decoding framework, to the best of our knowledge, which is the first to apply the CTC algorithm within the speculative decoding domain. This approach can not only generate drafts in a non-autoregressive way but also introduce correlations between draft tokens through probability allocation.

2. Through experiments conducted on MT-bench and GSM8K using various LLMs as base models, we have demonstrated superior speedup ability of the CTC-based draft model compared to other speculative decoding improvement methods. These results prove the rationality and effectiveness of our method.

2 Background

Recent advancements have emerged from the innovative approach of Blockwise Decoding[17], which introduced the draft-then-verify paradigm, leading to the development of Speculative Decoding[12] and Speculative Sampling[4]. These methodologies offer promising avenues for enhancing the speed of Large Language Models (LLMs).

Speculative Decoding predicts multiple future tokens and verifies their accuracy within a single decoding step. Using greedy sampling as an illustration: at step t, given an initial prompt X and previously produced tokens $y_{<t} = y_1, ..., y_{t-1}$, a speculative sequence of length n, $y'_t, ..., y'_{t+n}$, is generated by the draft model with respective probabilities $p'_t, ..., p'_{t+n}$. The target LLM then computes the accurate probabilities $p_t, ..., p_{t+n}$ in one pass during verification. Each token y'_i is evaluated in sequence, with its acceptance probability given by $\min(1, p'_t/p_t)$. Upon rejection of a token y'_i, subsequent tokens are disregarded, and the rejected token is re-sampled using the adjusted distribution $P(y_i) = \text{norm}(\max(0, P(y_i|y_{<i}, X) - P'(y_i|y_{<i}, X)))$.

The effectiveness of Speculative Decoding significantly depends on designing an intelligent draft model for precise token prediction and devising an optimal strategy for token sequence verification[24]. Consequently, current research efforts concentrate on these aspects to further exploit the potential of Speculative Decoding for speed acceleration.

2.1 Design of Draft Model

Many researchers have pursued the strategy of designing a draft model that operates independently of the base model[15, 19, 5, 13], such as employing a non-autoregressive transformer for simultaneous token drafting[23]. To ensure compatibility with the base model, works like [12] opt for draft models with fewer parameters from the same model series.

However, independent draft models necessitate training or fine-tuning, posing flexibility issues when transitioning between base models. Alternatively, some approaches rely on modifying the base model itself for token drafting through moderate adjustments[3, 25, 26, 10]. For instance, [3] introduces an additional module comprised of linear layers atop the target LLM for drafting tokens independently for different positions. In contrast, [25] incorporates a bypass within the LLM, allowing for earlier exits during the model's layer-by-layer computation.

2.2 Optimization of Verification

The strategy for compiling drafted tokens into candidate sequences and the criteria for sequence selection are vital during the verification stage. Initially, forming a single candidate sequence from the most probable tokens across positions was the prevalent approach[16, 18]. To incorporate a broader range of draft sequences, SpecInfer[14] organizes draft tokens into a tree structure, with paths from the root to leaf nodes representing different candidate sequences. Regarding selection criteria, early methods only accepted sequences matching the target model's greedy decoding output[23]. Later, [4] introduced Nucleus Sampling as a more effective yet complex acceptance criterion.

2.3 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is tailored for sequence prediction tasks, especially applicable in speech and handwriting recognition[9]. CTC expands the output space, $\mathcal{Y}$, by introducing a blank token ϵ denoting 'output nothing', creating an augmented space $\mathcal{Y}^*$. It defines a function $\beta(y)$ that maps any sample $y \in \mathcal{Y}$ to a subset of $\mathcal{Y}^*$, containing all valid alignments. Conversely, β^{-1} processes alignments from $\mathcal{Y}^*$ back to $\mathcal{Y}$ by merging adjacent, repeated tokens and removing blanks, resulting in the target sentence.

The sequence-level CTC loss function offers superior context modeling capabilities compared to token-level alternatives and effectively manages variable-length outputs without necessitating alignment during training. The training objective leverages dynamic programming to aggregate over all potential alignments $a \in \mathcal{Y}^*$.

$$\log p(y) = \log \sum_{a \in \beta y} p(a) \tag{1}$$

For inference, the model generates alignment a, from which repeated tokens and blanks are removed to yield the final output $y = \beta^{-1}(a)$.

3 CTC-drafter Model

In this section, we describe our implementation of the proposed model. CTC-drafter improves the acceptance rate of draft tokens while keeps extra investment of draft time in a reasonable duration, consequently achieving superior inference speedup. We first give an illustration of CTC-drafter's model structure, analyzing the functions of involved modules. Subsequently, we clarify CTC-drafter's training strategies and inference procedure.

3.1 Model Structure

Our CTC-drafter model structure are displayed in Figure 1. The training strategy is on the upper part and the inference process is on the bottom part. The base model on the left side are the LLM we desired to accelerate, which generally is composed of an embedding layer, multiple attention transformer layers and a output LM head that maps the hidden states to probability in vocabulary dimension.

For the draft procedure, we insert an attention draft module which take the hidden states outputted from base model as input and predict the probability distributions of draft tokens. Here hidden states

 https://doi.org/10.52202/079017-2923

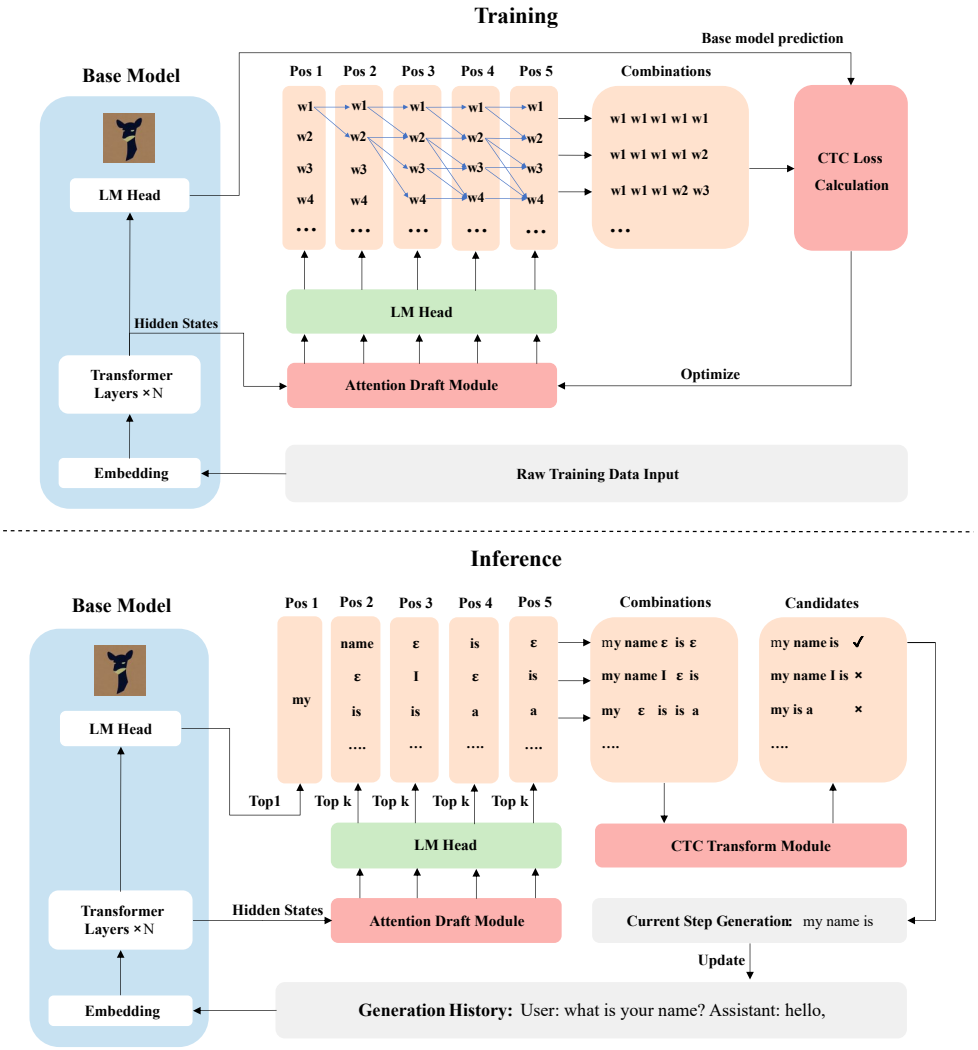

Figure 1: Illustration of CTC-drafter model training and inference strategy.

represents the compressed feature sequence after multiple transformer layers. Inspired by [13], the inner construction of Attention Draft Module is similar to the base model, with one single transformer layer conducting prediction in parallel.

For the verify procedure, raw candidate sequences are acquired by combining draft tokens in different places. Different from Medusa[3] which cuts off a part of combinations as prearranged, all the raw sequences keep the same length, containing possible draft tokens in each place. A CTC Transform Module is designed to process all raw sequences. The module first removes consecutive duplicate tokens and blank character, namely ε in Figure 1. Then the attention map that is used in base model verification calculation is modified. Positions in the attention map that corresponds to tokens been removed in CTC transform will be masked.

3.2 Training

The basic training strategy of CTC-drafter is displayed in Figure 1. We fixed the parameters of base model and trained the transformer layer in Attention Draft Module on ShareGPT dataset, which

is a subset of Vicuna's[6] training data. All input sequences are padded to the same max length, which is a predefined hyper parameter. Instead of conventional token-level cross entropy function that separately calculate the loss of each position, we use sequence-level CTC loss as the training objective. Note the input sequence as X and its corresponding label as Y, The dataset D contains a set of (X, Y) pair. The parameters of the trained draft model is noted as θ. Our training objective is optimizing θ to maximize the probability of labels across the dataset:

$$\theta = \underset{\theta}{\text{argmax}}(\sum_{(X,Y)\in D} P(Y|X,\theta)) \tag{2}$$

Conventionally, the training labels can be acquired by simply shifting the input. To train CTC-drafter, we follow the knowledge distillation method to calculate $Y_{distill}$ as labels Y in equation 2 by inputting base model the origin data [28], for the consideration that draft model can better match base model if trained on distilled dataset. First, base model outputs the probability distribution $P_{distill}(Y|X)$ in equation 3 and equation 4 trough multiple transformer layers and LM head, then we get distilled label sentence $Y_{distill}$ by greedy decoding in equation 5:

$$F_{distill} = \text{BaseModel}(X) \tag{3}$$

$$P_{distill}(Y|X) = \text{Softmax}(\text{LmHead}(F_{distill})) \tag{4}$$

$$\text{Y}_{distill} = \underset{Y}{\text{argmax}}(P_{distill}(Y|X)) \tag{5}$$

Given labels, we use sequence-level CTC loss to model equation 2. Assume the set $A_{X,Y}$ contains all possible raw sequences A that can be converted to Y after removing reduplicate tokens and blank characters. Sum up all probabilities of sequence in the set to get $P(Y|X,\theta)$. Considering the time complexity, it is impractical to enumerate all sequence. In Figure 1, we briefly show the dynamic programming of traversing all routes to calculate training labels probability $P(Y|X,\theta)$ in equation 6. The detailed algorithm is discussed in [9].

$$P(Y|X,\theta) = \sum_{A\in A_{(X,Y)}} P(A|X,\theta) \tag{6}$$

Further, the probability of each sequence $P(A|X,\theta)$ equals to the product of probability of each token a_i in the sequence with the independent assumption:

$$P(A|X,\theta) = \prod_{t=1}^{T} p(a_t|X,\theta) \tag{7}$$

$$A = (a_1, a_2, a_3, \ldots, a_t, a_{t+1}, \ldots, a_n) \tag{8}$$

The probability distribution of each token in the sentence is predicted by the transformer layer we added in draft module based on the hidden states outputted from base model as in equation 3:

$$\hat{F} = \text{DraftModule}(F_{distill}) \tag{9}$$

$$\hat{P}(Y|X,\theta) = \text{Softmax}(\text{LmHead}(\hat{F})) \tag{10}$$

$\hat{P}(Y|X,\theta)$ actually contains a group of probability distribution of different places in the sentence. Locate the corresponding position $\hat{P}_t(Y|X,\theta)$ to calculate $p(a_t|X,\theta)$ in equation 7:

$$p(a_t|X,\theta) = \hat{P}_t(Y = a_t|X,\theta) \tag{11}$$

3.3 Inference

To clarify the inference speedup mechanism of CTC-drafter, we further explain this procedure in one specific decoding step with the decoding history "Usr: what is your name? Assistant: hello," as base model input. The input will first be passed through base model producing the hidden states and greedy sampling base token "my". Attention Draft Module takes the hidden states from last transformer layer as input, outputting probability distributions of different positions after base token after LM Head projection.

For every position, the top k tokens are selected in descending order of probability, where k is predefined. In this instance, Attention Draft Module suggests that "name" is the best candidate token

 https://doi.org/10.52202/079017-2923

Table 1: performance of average speedup ratio on MT-bench.γ represents the average speedup ratio for all evaluation questions relative to Vanilla method, calculated by equation 13. β represents the average number of accepted tokens per decoding step for all evaluation questions, calculated by equation 12.

Speculation Method	Vicuna-7B		Vicuna-13B		Vicuna-33B	
	γ	β	γ	β	γ	β
MT-bench						
Vanilla[6]	1.00×	1.00	1.00×	1.00	1.00×	1.00
Medusa[3]	2.13×	2.58	1.97×	2.60	1.93×	2.55
Hydra[2]	2.36×	3.04	2.17×	3.06	2.15×	2.95
CTC-drafter	**2.78×**	**3.56**	**2.52×**	**3.51**	**2.20×**	**3.53**
GSM8K						
Vanilla[6]	1.00×	1.00	1.00×	1.00	1.00×	1.00
Medusa[3]	2.33×	2.78	2.21×	2.68	2.10×	2.46
CTC-drafter	**2.43×**	**3.53**	**2.66×**	**3.53**	**2.16×**	**3.40**

right next to "My" while it is possible that a blank character appears after "my" and the first draft token. Attempting to cover more reasonable candidate sequences, tokens in each place with different probability are combined in token tree structure and a group of the most valuable combinations are reserved as the raw candidate sequences. The raw candidate sequences is refined in CTC Transform Module, removing repetitive tokens and blank character and modifying the attention map.

All candidates are verified parallelly in base model, the longest sequence that satisfies the criterion will be selected as current decoding step's output, which in this case is "my name is". The decoding history is updated according to the output. In one single decoding step, compared with autoregressive decoding which will only produces token "My", CTC-drafter enables multiple tokens to be outputted, thus reducing overall base model calculation steps and achieving speedup.

4 Experiments

4.1 Implementation Settings

We choose open-source Vicuna large language model[6] with different parameter sizes as base model to conduct experiments. Vicuna models is fine-tuned on ShareGPT dataset based on LLaMA model, which are noted below as Vicuna-7b, Vicuna-13b and Vicuna-33b according to different parameter sizes. We also conduct training on LLaMA-2-Chat base models, detailed in the Appendix.

We fix Vicuna model's parameters and train the transformer layer inside draft module on ShareGPT dataset. The learning rate is set to 3×10^{-5}. To avoid gradient explosion, we adopt gradient clipping, setting the clipping threshold to 0.5. We set the max length of training data to 2048. All training tasks were executed on four 24GB NVIDIA GeForce RTX 3090 devices, taking around two days. To fully utilize graphics memory and accelerate training, we load models with FP16 precision for quantization. For comparison, we also implemented Medusa[3] on Vicuna models, following suggested experiment settings and retrain on the same ShareGPT dataset.

Trained models are evaluated on MT-bench and GSM8K datasets to assess the acceleration performance in various scenarios. MT-Bench is a carefully curated benchmark that includes 80 high-quality, multi-turn questions covering 8 primary categories of user prompts such as writing, roleplay and extraction[27]. GSM8K contains 8.5K high quality linguistically diverse grade school math problems[7]. Unlike some other datasets that offer base model questions with definitive answers such as multiple-choice questions, the two selected evaluation datasets contain open-ended questions, requiring base model to output long sequence answers in multiple decoding steps.

For every question, we record the total number of tokens in its corresponding answer as N, the total inference time T and the base model decoding steps M. We calculate the average number of tokens accepted per decoding step and the inference speedup compared to vanilla base model without

Table 2: Performance of average speedup ratio on MT-bench for different model structures.γ represents the average speedup ratio for all evaluation questions relative to Vanilla method, calculated by equation 13. β represents the average number of accepted tokens per decoding step for all evaluation questions, calculated by equation 12.

	CTC Verify		Medusa verify	
Speculation method	γ	β	γ	β
Linear layer + Cross Entropy Loss	1.71×	2.38	2.13×	2.58
Transformer layer + CTC Loss	2.78×	3.56	2.25×	3.02

speculative decoding:

$$\text{Accepted tokens} = \frac{N}{M} \tag{12}$$

$$\text{speedup} = \frac{\bar{T}_{\text{vanilla}}}{\bar{T}_{\text{spec}}} = \frac{T_{\text{vanilla}}/N_{\text{vanilla}}}{T_{\text{spec}}/N_{\text{spec}}} \tag{13}$$

The average number of accepted tokens reflects models' ability to speculate candidate tokens. However, it takes extra inference time to draft tokens. Thus, the speedup metrics can be viewed as a trade-off between speculation quality and extra time consumed.

4.2 Results and analysis

The performance of different speculation methods on MT-bench and GSM8K are showed in Table 1. The speedup ratio of Medusa is evaluated following recommended setting in its corresponding technical report. The results of Hydra[2] on Vicuna models are acquired from its corresponding paper. We also measures the performance of fully auto-regressive decoding with no speculation method as baseline to calculate speedup ratio of other three methods, noted as Vanilla in Table 1.

MT-bench. The results show that our proposed CTC-drafter achieves better draft quality on MT-bench compared other works, with more than three tokens been accepted per decoding steps. Higher predicting accuracy enables CTC-drafter to achieve speedup ratio of more than 2×, outperforming Medusa and Hydra on all types of base model. Besides, the speedup performance of all speculation method is influenced as base model size increases. Possible explanation is that we can not significantly expand the size of draft module considering extra time consumed, when base model size increase, larger ability gap between base model and draft module makes it more difficult for draft module to imitate base model's prediction behavior. Therefore, the average number of accept tokens β of CTC-drafter decreases from 3.56 to 3.53, influencing the speedup performance.

GSM8K. Compared with MT-bench, questions in GSM8K mainly focus on math category. As is shown on the bottom of Table 1, our proposed CTC-drafter keep a superior speedup performance over Medusa for all base models. Besides, the speedup performance suffers to some extent for CTC-drafter in Vicuna-7B base model, compared with the performance in MT-bench. The main reason is that we completely rely on the comprehensive ability of origin base model to offer answers without fine-tuning on GSM8K training dataset. Fortunately, when the base model size increases to 13B, CTC-drafter maintains prediction accuracy, achieving 2.66× speedup. However, bridging the capability gap for Vicuna-33B is challenging, leading to a decline in performance.

4.3 Ablation experiments

In this part, we list the results of ablation experiments and further analyze the working paradigm of CTC-drafter. First, we explore how each part of the model structure influences the acceleration performance in Table 2. Then we illustrate how the prediction ability varies across different categories of test questions in Figure 2. Besides, to better visualize the trade-off between extra time consumption and prediction accuracy, the time consumption of each calculation procedure during inference is measured in Figure 3.

Model structure. To better utilize the context information, CTC loss is used as the training objective as discussed in Section 3.2, which optimizes the draft module under sequence-level supervision. Besides, we replace the linear layers of Medusa head with more complex transformer layer to suit

 https://doi.org/10.52202/079017-2923

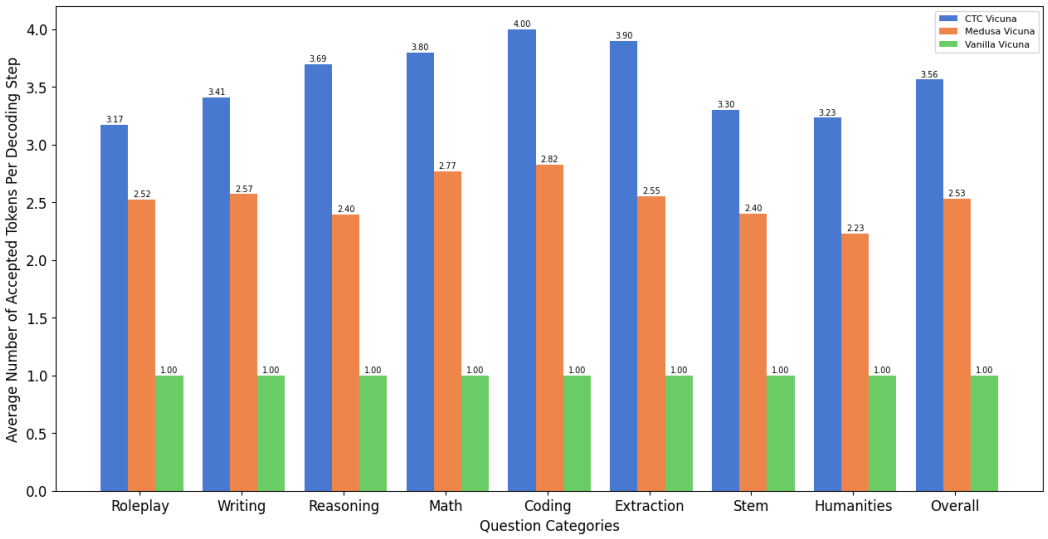

Figure 2: Average number of tokens accepted per decoding step in different question categories on MT-bench, with Vicuna-7B as base model. The performance on Vicuna-13B and Vicuna-33B is consistent with this result. The blue color represents CTC-drafter method, orange color represents Medusa method and green color represents baseline. All evaluation experiments are conducted on the same device.

CTC loss and better imitate the base model. For inference, the original token tree verification strategy is modified to include extra operations such as CTC transform and attention map modification.

To explore the speedup contribution of each part of modification as discussed above, we replace the draft module and verify module with the corresponding ones in Medusa and conduct experiments on the modified speculation methods. The results are showed in Table 2, with Vicuna-7B as base model. In this table, linear layer represents drafting tokens based on linear layers with cross-entropy Loss function as training objective, which is adopted by Medusa. Transformer layer represents drafting tokens based on transformer layers with CTC loss as training objective, which is adopted by CTC-drafter. Medusa verify refers to vanilla token tree verification described in [14]. CTC verify includes extra operations compared with Medusa verify including CTC transform of candidate sequences and attention map modification as mentioned before.

Replacing linear layer with transformer layer and using CTC loss to design training loss function increase the average number of accepted tokens β from 2.58 to 3.02. It is clear that with these two efforts combined, draft module is guided to conduct attention across the whole input sentence instead of simply learning offsets of the last hidden states. However, blank characters and repeated tokens exist in the candidate sequences spoil draft quality and speedup for models without CTC transform, β decreases from 3.56 to 3.02 and γ from 2.78× to 2.25×.

Question categories. We further explore how speedup performance varies on different question categories, showing in Figure 2. Both CTC-drafter and Medusa achieved the best prediction accuracy on coding category, which can be attributed to the highly logical nature of the problems within this category. Among all categories, the acceptance rate of roleplay questions is slightly low for CTC-drafter, which may be due to the deficiency of questions of this category in our training datasets.

Time consumption. Compared with Medusa, it is unavoidable that our methods' draft strategy requires more complex calculations. We display each stage's time consumption throughout the whole inference decoding process in Figure 3. First, we replace the original medusa head with transformer layer to better fit the base model, which cause the time of draft model increases from 3.71% to 14.93%. Besides, for the need to dynamically process candidate sequences in each decoding round, the CTC transform accounts for extra 5.36% of the overall decoding time consumption. Considering that the base model's calculation still account for the main part, it is acceptable that we increase the draft ability and thus reducing base model's decoding rounds, which balances the extra time consumption and achieve better speedup on the whole.

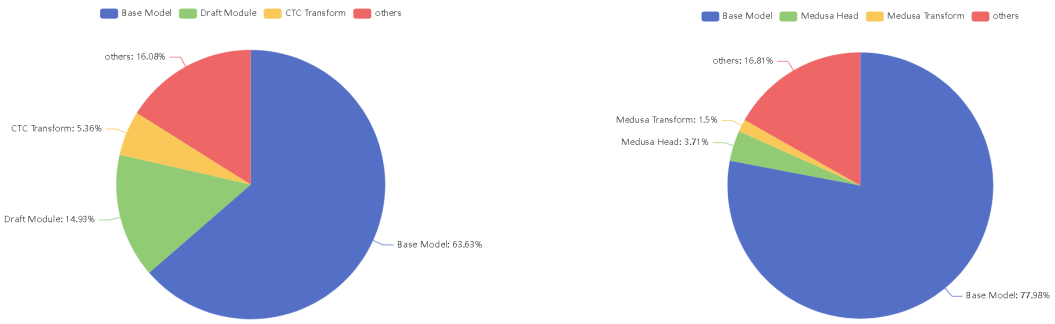

Figure 3: The percentage of time consumed for different processes based on CTC-drafter(left) and Medusa(right) speculation strategies. The "others" part mainly contains matrix operations involved in token tree verification.

5 Related Work

Medusa. After Speculative decoding[23] and Speculative sampling[12], many improvement works has been proposed to optimize the draft model and verification strategy. Among these, Medusa explores a novel and efficient acceleration framework[3]. Instead of using an independent model with fewer parameters as the draft model, several Medusa Heads are added on top of the last transformer layer of base model. The i-th Medusa Head is responsible for predicting the i-th token after the base model decoding token in each step. For each Medusa Head, top k tokens with highest probability are selected and combined in tree structure to form candidate sequences. Using tree mask method in [14], the multiple sequences are validated in parallel during the next decoding step. Two strategies are used for the training of Medusa Head: Medusa-1 fix the parameters of base model, optimize only the Medusa Head on a small dataset, Medusa-2 adopt end-to-end training, fine-tuning base model and Medusa Head together on larger datasets. Although Medusa-2 can achieve remarkable inference speedup, the need of large training data and time consumption limits its generality across different base model. In this paper, we only implement and demonstrate Medusa-1 considering fairness.

Hydra. Modified from Medusa Head, Hydra designs Hydra Head, a sequentially dependent, drop-in replacement for standard draft heads[2]. A Hydra Head conduct prediction not only based on base model hidden states but also the decoding output of other Hydra Heads, which significantly improves speculation accuracy. Besides, some other tricks are adopted for further inference speedup including adding noise and knowledge distillation.

6 Conclusion and Future work

Speculation decoding and corresponding improvement works mostly draft candidate tokens without considering context information and generate fixed-length candidate sequences for verification, which not only influences the draft quality, bur also lacks generality across different large language models. In this paper, we propose a novel framework named CTC-drafter based on CTC algorithm. Specifically, we use CTC loss as the training objective to model the context connection instead of cross entropy. We reconstruct the structure of draft model, using transformer layer to better fit base models. Besides, with CTC transform, we achieve adaptive candidate sequence generation which makes it convenient to transfer the framework across different base models. Nevertheless, our current work is subject to certain limitations that requires careful consideration. More training tricks can be explored to further enhance the prediction ability of draft module. Besides, it is still doubtful that whether the current draft model structure is optimal. What's more, different types of large pretrained language model need to be adopted in our proposed framework to evaluate the acceleration performance of CTC-based draft model.

For the future work, we attempt to identify techniques to reduce the extra time consumed caused by more complex draft operations introduced. Other verification criteria such as Nuclear Sampling [4] can be integrated into our framework. What's more, some other methods such as Conditional Random Field(CRF, [20]) and Directed Acyclic Graph(DAG, [8]) can be explored to model the context information when drafting tokens, we remain these ideas for future work.

https://doi.org/10.52202/079017-2923

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

[3] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. Medusa: Simple framework for accelerating llm generation with multiple decoding heads, 2023.

[4] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[5] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Jie Huang, and Kevin Chen-Chuan Chang. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*, 2023.

[6] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Jean C Digitale, Jeffrey N Martin, and Medellena Maria Glymour. Tutorial on directed acyclic graphs. *Journal of Clinical Epidemiology*, 142:264–267, 2022.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[10] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Sophia Shao. Speed: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.

[11] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

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

[13] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[14] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 1(2):4, 2023.

[15] Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

[16] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR, 2019.

[17] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[18] Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*, 2021.

[19] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

[21] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[22] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[23] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, 2023.

[24] Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

[25] Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*, 2023.

[26] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.

[27] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

 https://doi.org/10.52202/079017-2923

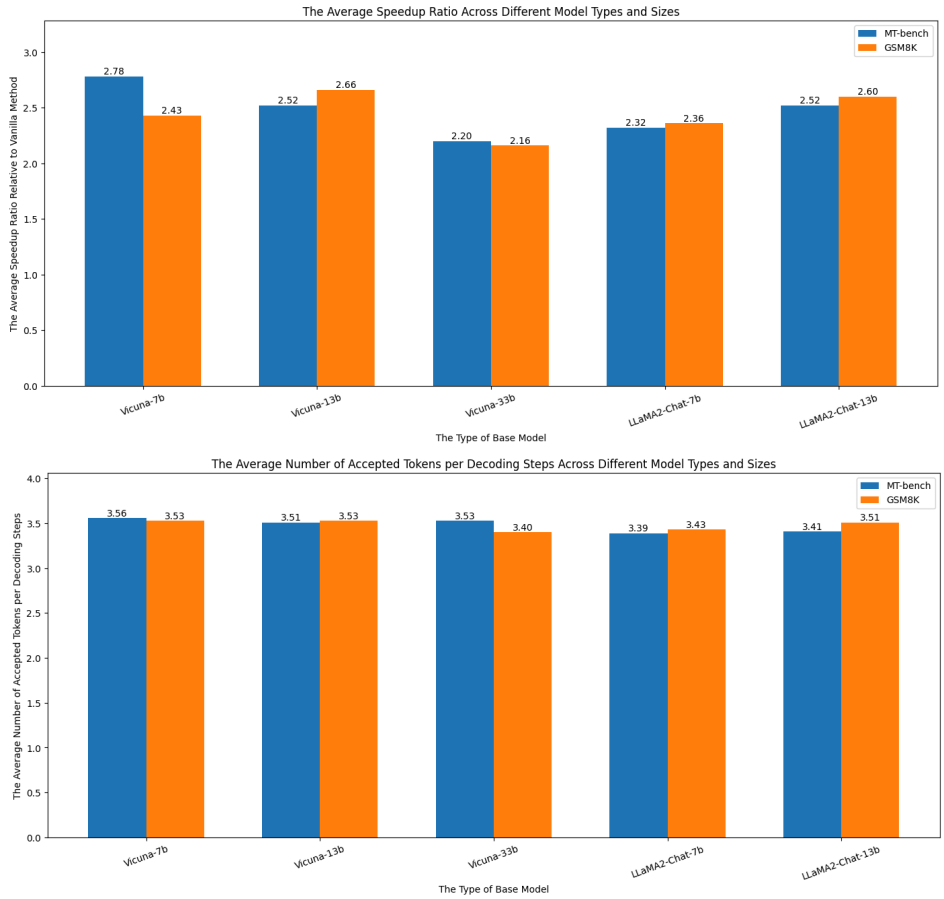

Figure 4: The bar charts of speedup ratio relative to vanilla method γ(top) and average number of tokens accepted per decoding step β(bottom) across different model types and sizes with CTC-drafter. The blue bar represents performance on MT-bench and the orange bar represents GSM8K. All evaluation experiments are conducted on the same devices.

A Appendix

To evaluate the generality of our method across different base models, we add supplementary experiments on LLaMA-2-Chat base models[22]. We select LLaMA-2-Chat 7b and 13b as base models and evaluate CTC-drafter's performance on MT-bench and GSM8K. For a clear comparison, the evaluation results of various base models, including Vicuna, are documented in Figure 4.

CTC-drafter maintains ideal performance when transferring from Vicuna models to LLaMA-2-Chat models, only slight decline when compared the evaluation results on Vicuna-7b and LLaMA-2-chat-7b. Besides, it should be noted that increasing the size of the LLaMA-2-Chat model to 13b does not compromise draft quality, while enhancing speedup performance. This trend diverges from Vicuna base models, potentially due to distinct inference paradigms inherent in both models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In Section 3(CTC-drafter) we illustrate our contributions for novel acceleration framework design. In Section 4(Experiments) experimental results are showed to support our contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 6 we discuss the limitations of our work and leave these for future work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-2923

Answer: [Yes]

Justification: In Section 3.2, we declare the training objective with full set of assumptions and a complete (and correct) proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 4.1(Implementation Settings) we provide the main settings and steps to train the models and conduct evaluation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code in Supplementary Material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 4.1(Implementation Settings) we provide the main settings and steps to train the models and conduct evaluation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In section 4(Experiments), we discuss the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

https://doi.org/10.52202/079017-2923

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Section 4.1(Implementation Settings), we discuss the devices we use and time consumption.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In Section 1(Introduction) we discuss potential societal impacts of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: All pretrained language models and datasets we use is open-source without a high risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets we use are cited and credited properly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

https://doi.org/10.52202/079017-2923

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: the paper does not release new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-2923