GliDe with a CaPE: A Low-Hassle Method to Accelerate Speculative Decoding

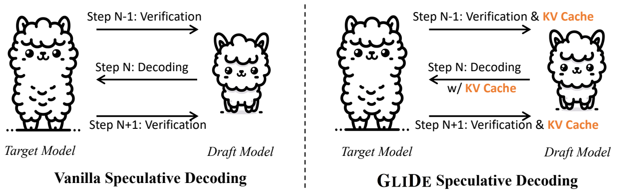
Cunxiao Du [1] **Jing Jiang** [1] **Xu Yuanchen** [2] **Jiawei Wu** [2] **Sicheng Yu** [1] **Yongqi Li** [3] **Shenggui Li** [4] **Kai Xu** [4]
Liqiang Nie [5] **Zhaopeng Tu** [6] **Yang You** [2]

Abstract

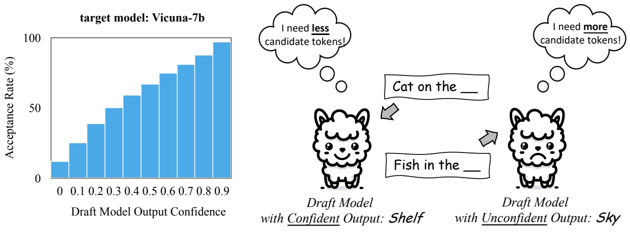
Speculative decoding is a relatively new decoding framework that leverages small and efficient draft models to reduce the latency of LLMs. In this study, we introduce GliDe and CaPE, two low-hassle modifications to vanilla speculative decoding to further improve the decoding speed of a frozen LLM. Specifically, GliDe is a modified draft model architecture that reuses the cached keys and values from the target LLM, while CaPE is a proposal expansion method that uses the draft model's confidence scores to help select additional candidate tokens for verification. Extensive experiments on different benchmarks demonstrate that our proposed GliDe draft model significantly reduces the expected decoding latency. Additional evaluation using walltime reveals that GliDe can accelerate Vicuna models up to 2.17x and further extend the improvement to 2.61x with CaPE. We release our code, data, and the trained draft models at `https://github.com/NonvolatileMemory/GliDe_with_a_CaPE_ICML_24`.

1. Introduction

Large language models (LLMs) have become increasingly powerful and are now adopted for a wide range of applications such as machine translation (Jiao et al., 2023) and fact verification (Zhang & Gao, 2023; 2024). Many LLM applications require real-time responses, e.g., machine translation systems (Yu et al., 2022b). However, LLMs are typically based on the autoregressive transformer architecture, which generates output tokens step by step and thus suffers from high latency, particularly with larger model sizes. To reduce LLM serving latency, *speculative decoding* (SD) has been proposed as a viable solution (Stern et al., 2018; Chen et al., 2023; Leviathan et al., 2023). The key idea of SD is to use a smaller and more efficient *draft* model to predict the next γ tokens in the output sequence and then use the original LLM, which is called the *target* model, to verify the proposed γ tokens *in parallel*. SD hinges on the insight that not all output tokens are equally difficult to predict; employing a smaller but more efficient draft model that can correctly predict those "easy" tokens helps save inference time. The success of SD relies heavily on the acceptance rate of the tokens proposed by the draft model, or in other words, how likely the target model will accept the proposed tokens. Therefore, the design of the draft model and its proposal mechanism play a central role in accelerating SD.



(a) How Speculative with KV Cache Decoding works?



(b) Confidence v.s. Acceptance Rate, how to utilize it?

Figure 1. (a) Comparison between vanilla speculative decoding and our GliDe. (b) There is a positive correlation between acceptance rate and confidence score.

Recent work on SD acceleration has explored two promising directions. The first direction relies on the intuition that the more closely the draft model's predictions align with those of the target model, the greater the likelihood of the proposed tokens being accepted. Along this line, researchers have proposed to align the draft model with the target model through distillation training (Zhou et al., 2024; Miao et al., 2023; Liu et al., 2023) or to use the

---

[1]Singapore Management University [2]National University of Singapore [3]The Hong Kong Polytechnic University [4]HPC-AI Tech. [5]Harbin Institute of Technology (Shenzhen) [6]Tencent AI Lab. Correspondence to: Jing Jiang <jingjiang@smu.edu.sg>, Zhaopeng Tu <zptu@tencent.com>, Yang You <youy@comp.nus.edu.sg>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

same model for both speculation and verification (i.e., self-speculative decoding) (Fu et al., 2023). Second, at each round of speculation, one can ask draft models to propose multiple candidate sequences for the target model to verify, thereby increasing the chance of acceptance. In this vein, recent work explores employing multiple draft models to propose different candidate token sequences (Miao et al., 2023) and using multiple prediction heads, one for each position, to propose multiple candidate token sequences in a non-autoregressive manner (Cai et al., 2023).

In this work, we propose new solutions in the two aforementioned directions that are easy to implement and yet highly effective for accelerating SD. Our solutions are motivated by the following insights. First, we make an important observation that the draft model does not need to work separately from the target model during inference; by leveraging the KV (key-value) cache of the target model, the draft model can propose tokens that are more likely to be accepted by the target model. Second, we note that previous attempts to propose multiple candidate sequences are not ideal; they either rely on multiple draft models (Miao et al., 2023), which is not always feasible in practice, or use non-autoregressive decoding to keep the draft model's inference time low but sacrifice the fluency of the proposed candidate sequences (Cai et al., 2023). We propose a new solution that, without imposing much additional computational or memory requirements, simply expands a single candidate sequence with additional candidate tokens at each position.

Specifically, we introduce a cross-attention layer in the draft model's network architecture. Although its implementation appears straightforward, this layer enables the draft model to access the target model's KV cache, a process that is empirically proven to be highly effective in our experiments. We metaphorically describe this cross-attention mechanism as "catching a glimpse" of the KV cache and name our model GLIDE, an acronym for **Gl**impse **D**raft Mod**e**l. The idea is illustrated in Figure 1(a). As we can see from the figure, because the target model stores its computed keys and values corresponding to the tokens it has verified in the last round of verification, these KV cache entries are free for the draft model to use. By re-using these keys and values, the draft model is more likely to behave similarly to the target model.

To expand a proposed candidate sequence with additional candidate tokens, a key question is how many additional top-ranked tokens we should include at each position. Although a naive solution is to use a pre-defined and fixed number, we suspect that we should provide more candidate tokens at a position where the top-1 candidate token has a low confidence score. To verify whether this hypothesis holds, we first conduct a preliminary experiment to check whether a proposed token's prediction confidence is correlated with the chance of its acceptance. Figure 1(b) shows the experiment results. We can see that indeed there is a clear positive correlation between the draft model's prediction confidence of a proposed token and the token's chance of being accepted by the target model. This observed correlation motivates us to design a **C**onfidence-**a**ware **P**roposal **E**xpansion (CAPE) method that dynamically determines the number of additional candidate tokens to include in an expanded proposal sequence.

We conduct extensive experiments to evaluate the effectiveness of GLIDE and CAPE. Experiments on four datasets using Vicuna and Mistral as target models show that it is highly effective for GLIDE to attend to the target model's KV cache, leading to an improvement of the acceptance rate of up to 23.5%, and GLIDE significantly outperforms several baseline draft models, with an average improvement of 19.9% in terms of acceptance rate, compared with previous draft models. Additional experiments incorporating CAPE into GLIDE reveal that CAPE can achieve speedups ranging from 2.50x to 2.61x on different Vicuna models (Chiang et al., 2023) based on wall-time. The code and pretrained draft models can be found in `https://github.com/NonvolatileMemory/GliDe_with_a_CaPE_ICML_24`.

2. Related Work

Speculative decoding (Xia et al., 2024) was first proposed as block-wise parallel decoding. Stern et al. (2018) trained multiple auxiliary models to predict γ future tokens in parallel and then used the original model to verify the future tokens in parallel. Inspired by this idea, Chen et al. (2023); Kim et al. (2023); Yuan et al. (2023); Leviathan et al. (2023); Xia et al. (2023) used an independent draft model to propose a short sequence of tokens for the target model to verify. There were also draft model-free works (He et al., 2023; Yang et al., 2023; Fu et al., 2023; Zhang et al., 2023) for speculative decoding.

Leviathan et al. (2023) pointed out that the main factors for speedup are the efficiency of the draft model and the acceptance rate of the proposed tokens. To improve the efficiency of the draft model, Cai et al. (2023) proposed adding multiple LM Heads to predict future tokens at different positions independently. To improve the acceptance rate, Miao et al. (2023) proposed to generate multiple candidate sequences, which can be efficiently verified through a tree verification process (Miao et al., 2023; Cai et al., 2023; Sun et al., 2023). Alternatively, distillation was used to train draft models that hopefully are similar to the target model. Zhou et al. (2024) proposed using sequence-level distillation (Kim & Rush, 2016) via the draft model's output, while Liu et al. (2023) resorted to online distillation to quickly adapt the draft model to the current context.

Our work focuses on the second factor, i.e., acceleration through improving the acceptance rate. GLIDE re-uses the target model's KV cache to generate proposals more likely to be accepted, and CAPE expands the proposals with additional highly-ranked candidate tokens to further increase the acceptance rate.

3. Background: Speculative Decoding (SD)

In vanilla SD, we assume that there is a powerful but slow LLM (i.e., the *target* model, denoted as $\mathcal{M}_{\text{T}}$) used to verify the final output sequence. Meanwhile, a less powerful but faster language model (i.e., the *draft* model, denoted as $\mathcal{M}_{\text{D}}$) is used to propose candidate tokens. Let $x_{\leq t}$ denote the prefix or prompt from which the next tokens are to be proposed. SD works by first using the faster draft model $\mathcal{M}_{\text{D}}$ to autoregressively generate the next γ tokens $(x_{t+1}, \ldots, x_{t+\gamma})$. We refer to this proposed candidate token sequence as a *proposal*. However, because the proposal may not be what the target model $\mathcal{M}_{\text{T}}$ would have generated, the proposed tokens need to be *verified* by $\mathcal{M}_{\text{T}}$. $\mathcal{M}_{\text{T}}$ can verify the γ tokens in parallel, which is much faster than autoregressive generation. The verification step returns the first n tokens ($0 \leq n \leq \gamma$) in the proposal that are accepted by $\mathcal{M}_{\text{T}}$ based on some acceptance strategy. In addition, the verification step returns one more token x'_{t+n+1} for free. Then the sequence $(x_{t+1}, \ldots, x_{t+n}, x'_{t+n+1})$ is appended to the original prefix $x_{\leq t}$ to form the new prefix for the next round of speculation and verification. There are two acceptance strategies: *speculative decoding* and *speculative sampling*. Leviathan et al. (2023) show that the outputs of these two strategies are equivalent to the outputs of the target model with greedy search and random sampling decoding strategies, respectively.

4. Our Method: GLIDE with a CAPE

As stated in §1, our method to accelerate SD consists of two parts: (1) We design a draft model called GLIDE that "catches a glimpse of" the KV cache computed during the *target model's last round of verification* to assist the draft model's current round of proposal generation. (2) We propose a proposal expansion mechanism called CAPE that uses the confidence scores of the draft model to dynamically determine how many additional top-ranked tokens to include at each position in a proposal sequence.

4.1. GLIDE: Glimpse Draft Model

We assume that both the target and the draft models follow the standard decoder-only transformer architecture. To enable the draft model to take advantage of the hidden representations of the prefix tokens computed and cached by the target model, we propose a new architecture for the draft model called GLIDE (Glimpse Draft Model). GLIDE allows the draft model $\mathcal{M}_{\text{D}}$ to re-use the cached key-value pairs in the target model $\mathcal{M}_{\text{T}}$, presumably making the distribution of $\mathcal{M}_{\text{D}}$ more consistent with that of $\mathcal{M}_{\text{T}}$ without incurring much additional computational cost. Our $\mathcal{M}_{\text{D}}$ will be trained from scratch based on the GLIDE architecture while $\mathcal{M}_{\text{T}}$ is kept frozen.

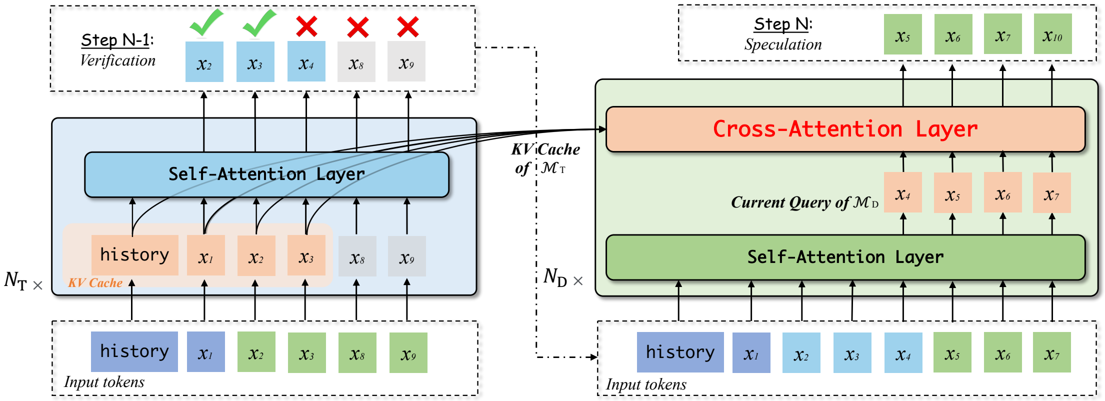
Specifically, assume that in the last round of speculation, $\mathcal{M}_{\text{D}}$ has proposed a sequence of tokens that is passed to $\mathcal{M}_{\text{T}}$ for verification. Then in the subsequent verification step, assume that $\mathcal{M}_{\text{T}}$ accepts the sequence up to x_{t-1} and generates an additional token x_t. After this verification step, we will keep only the KV cache for tokens up to position $(t-1)$ and discard the KV cache for those tokens rejected by $\mathcal{M}_{\text{T}}$. The left hand side of Figure 2 illustrates the verification process, where the grey cube represents the discarded KV cache.

Now assume that given the prefix $x_{\leq t}$, the draft model $\mathcal{M}_{\text{D}}$ in its current round of speculation has proposed tokens x_{t+1} to x_{t+i-1}. Next, $\mathcal{M}_{\text{D}}$ is going to propose the next token x_{t+i} based on the prefix $x_{<(t+i)}$. In vanilla speculative decoding, $\mathcal{M}_{\text{D}}$ works independently of $\mathcal{M}_{\text{T}}$. In our proposed GLIDE architecture, however, we want to re-use $\mathcal{M}_{\text{T}}$'s cached keys and values associated with the prefix tokens up to x_{t-1}. To do so, we insert a cross-attention sub-layer between the self-attention sub-layer and the feed-forward sub-layer in each transformer layer of $\mathcal{M}_{\text{D}}$, as shown in the right hand side of Figure 2. This additional layer first projects the outputs between t and $(t+i-1)$ from the self-attention sub-layer below into query vectors of a dimension that is compatible with $\mathcal{M}_{\text{T}}$'s keys and values. Then this cross-attention sub-layer performs standard cross-attention between these queries from $\mathcal{M}_{\text{D}}$ and the KV cache from $\mathcal{M}_{\text{T}}$. The resulting vectors will be passed to the feed-forward sub-layer above.

Concretely, let h denote the number of heads in each self-attention sub-layer of $\mathcal{M}_{\text{T}}$. Let $(K^{l,j}, V^{l,j})$ denote the keys and values of the j-th head at the l-th layer of $\mathcal{M}_{\text{T}}$, where $K^{l,j}, V^{l,j} \in \mathbb{R}^{(t-1)\times d_k}$. Let $H^m \in \mathbb{R}^{i \times d_{\text{D}}}$ represent the last i output vectors from the m-th self-attention sub-layer of $\mathcal{M}_{\text{D}}$. These i vectors in H^m correspond to positions t to $(t+i-1)$. We intend to use these vectors from $\mathcal{M}_{\text{D}}$ as queries to attend to the representations of the prefix tokens up to x_{t-1} in $\mathcal{M}_{\text{T}}$. Because d_{D} (the dimension of the hidden vectors in $\mathcal{M}_{\text{D}}$) is generally different from d_k (the dimension of the keys in $\mathcal{M}_{\text{T}}$), and because $\mathcal{M}_{\text{T}}$ has h attention heads, we first perform h linear projections to project H^m into h different query matrices, one for each head in $\mathcal{M}_{\text{T}}$:

$$Q^{m,j} = H^m W_j,$$

where $j \in [1, h]$, and $W_j \in \mathbb{R}^{d_{\text{D}} \times d_k}$ are learnable parameters. To perform cross-attention, each layer in $\mathcal{M}_{\text{D}}$ attends


Step N-1:
Verification
✓ ✓ ✗ ✗ ✗
x_2 x_3 x_4 x_8 x_9
Self-Attention Layer
history x_1 x_2 x_3 x_8 x_9
KV Cache
N_T ×
history x_1 x_2 x_3 x_8 x_9
Input tokens
KV Cache of $\mathcal{M}_T$
Step N:
Speculation
x_5 x_6 x_7 x_{10}
Cross-Attention Layer
Current Query of $\mathcal{M}_D$
x_4 x_5 x_6 x_7
Self-Attention Layer
N_D ×
history x_1 x_2 x_3 x_4 x_5 x_6 x_7
Input tokens

$\mathcal{M}_T$ does parallel verification [x_2, x_3, x_8, x_9] from Step N-2, and delivers **KV Cache** for Step N speculation.

$\mathcal{M}_D$ autoregressively proposes next 4 tokens [x_5, x_6, x_7, x_{10}] for Step N+1 verification.

Figure 2. Overview of GLIDE. We insert the **cross-attention** layer in the Draft Model ($\mathcal{M}_D$) and let it glimpse the KV cache of the target model ($\mathcal{M}_T$) from the last verification.

to a corresponding layer in $\mathcal{M}_\text{T}$, counting from the top layers. In this way, we try to use the KV cache from the upper layers of $\mathcal{M}_\text{T}$ because presumably they are more contextualized and therefore better representations. Specifically, let N_T and N_D denote the numbers of layers of $\mathcal{M}_\text{T}$ and $\mathcal{M}_\text{D}$, respectively. For the m-th layer in $\mathcal{M}_\text{D}$, we let it attend to the l-th layer in $\mathcal{M}_\text{T}$, where $l = N_\text{T} - N_\text{D} + m$, through multi-head attention as follows:

$$\begin{aligned}\text{MultiHead}(Q^m, K^l, V^l) &= \text{Concat}(\text{hd}_1, \dots, \text{hd}_h)W^O,\\ \text{where hd}_j &= \text{SoftMax}(\frac{Q^{m,j}(K^{l,j})^\intercal}{\sqrt{d_k}})V^{l,j}.\end{aligned}$$

Here $W^O \in \mathbb{R}^{hd_k \times d_\text{D}}$ is a standard learnable parameter matrix that projects the concatenated heads back to dimension d_D. MultiHead(Q^m, K^l, V^l) will then be concatenated with the other output vectors (up to position t) from the m-th self-attention sub-layer of $\mathcal{M}_\text{D}$ and fed into the m-th feed-forward sub-layer of $\mathcal{M}_\text{D}$, as shown in Figure 2.

Although we could also attend to only the last layer of the KV cache, our preliminary experiments showed that using the last N_d layers gave slightly better performance. Because computation-wise attending to the last layer and attending to the last N_d layers do not have much difference, here we choose to attend to the last N_d layers of KV cache.

Block-wise Attention Mask. We train the draft model $\mathcal{M}_\text{D}$ from scratch similar to the standard training of autoregressive decoder-only models. A main difference is that during training, the target model $\mathcal{M}_\text{T}$ (which is kept frozen) is used to provide the KV cache for the cross-attention sub-layers of $\mathcal{M}_\text{D}$. However, care must be taken to ensure consistency between the training stage and the inference stage regarding the use of $\mathcal{M}_\text{T}$'s KV cache. Recall that at inference time when $\mathcal{M}_\text{D}$ is used for speculation, after $\mathcal{M}_\text{T}$ has verified tokens up to x_{t-1} and generated token x_t, and when $\mathcal{M}_\text{D}$ is about to speculate the $(t+i)$-th token, $\mathcal{M}_\text{D}$ only has access to the KV cache in $\mathcal{M}_\text{D}$ up to position $(t-1)$ rather than position $(t+i-1)$. This can be considered a KV cache delay. When training the draft model $\mathcal{M}_\text{D}$, we need to simulate this delayed KV cache to ensure that the trained $\mathcal{M}_\text{D}$ works well with delayed KV cache.

To do so, we introduce a training mechanism with a block-wise attention mask as follows. We divide the training sequences into blocks of length L (which is set to be 5 in our experiments). During training, when the draft model $\mathcal{M}_\text{D}$ is predicting token x_j that is in the i-th block of a sequence, in $\mathcal{M}_\text{D}$'s cross-attention sub-layer, we will use the representations of only the tokens in the i-th block to the left of x_j as queries. These queries will attend to the KV cache in $\mathcal{M}_\text{T}$ corresponding to the tokens in the first $(i-1)$ blocks of the sequence but not tokens in the i-th block. Formally, the block-wise attention mask takes an attention matrix A as input and is defined as follows:

$$\text{Mask}_\text{Block}(A_{jk}) = \begin{cases} A_{jk}, & \text{if block}(j) > \text{block}(k) \\ -\infty, & \text{otherwise} \end{cases},$$

where block(j) is a function that returns the index of the block where token x_j belongs. This attention mask is only used for the cross-attention sub-layers of the draft model during training and is not used for testing.

4.2. CAPE: Confidence-Aware Proposal Expansion

Our CAPE has two components: a proposal expansion mechanism and a corresponding verification mechanism. Figure 3 gives an overview of CAPE.

Proposal Expansion. A standard proposal generated by a draft model given prompt $x_{\leq t}$ consists of a sequence of γ

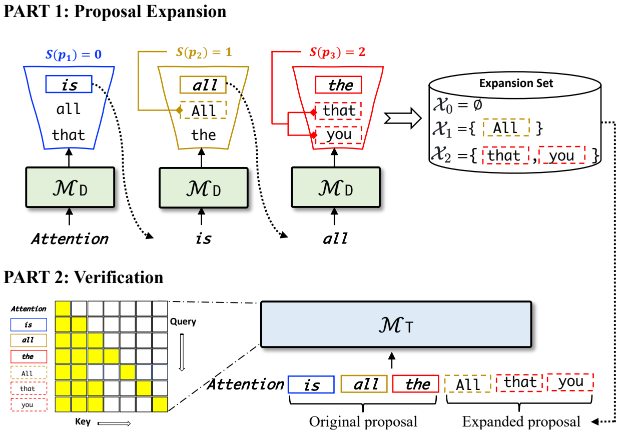


Figure 3. Overview of CAPE.

tokens, which we denote as $(x_{t+1}, x_{t+2}, \ldots, x_{t+\gamma})$. These tokens have the highest probabilities at their corresponding positions as predicted by the autoregressive draft model. In CAPE, we introduce a representation called *expanded proposal* that augments a standard proposal with a sequence of expansion sets, one for each token in the proposal. The expansion set $\mathcal{X}_i$ for x_{t+i} in the proposal contains other top-ranked tokens at position $(t+i)$. Formally, an expanded proposal is a sequence of tokens $(x_{t+1}, x_{t+2}, \ldots, x_{t+\gamma})$ where each token x_{t+i} is associated with an expansion set $\mathcal{X}_i = \{x'_{t+i,j}\}_{j=1}^{K_i}$ with each $x'_{t+i,j}$ being a token and K_i being the size of $\mathcal{X}_i$.

With minor modifications to the standard proposal generation procedure, an expanded proposal is constructed as follows. Let $p_{\mathrm{D}}(\cdot|x_{<t+i})$ denote the next token distribution computed by the draft model $\mathcal{M}_{\mathrm{D}}$. Similar to standard proposal generation, the token that has the highest probability according to $p_{\mathrm{D}}(\cdot|x_{<t+i})$ is selected to be x_{t+i} and appended to the proposal. Let $p_i = \max p_{\mathrm{D}}(\cdot|x_{<t+i})$ denote the probability (which can be regarded as a confidence score) of predicting this x_{t+i}. A function $S(\cdot)$ is then used to determine the size K_i of the expansion set $\mathcal{X}_i$ based on the confidence score p_i. As discussed in §1, when the confidence p_i is high, we expect the top-ranked x_{t+i} to be correct with a high chance and therefore will use a small K_i. When p_i is low, we will use a relatively large K_i. Concretely, we set $S(p)$ to be 7, 5, 3, and 1 for p in the ranges of $(0, 0.3]$, $(0.3, 0.6]$, $(0.6, 0.8]$, and $(0.8, 1]$, respectively. Once K_i is determined, we will select K_i tokens (excluding x_{t+i}) that have the highest probabilities according to $p_{\mathrm{D}}(\cdot|x_{<t+i})$ and place them in $\mathcal{X}_i$. These tokens are the next best choices for position $(t+i)$ other than x_{t+i}.

It is important to point out that identification of the highest-ranked tokens to be placed in $\mathcal{X}_i$ can be done while the draft model continues to autoregressively generate the following tokens beyond x_{t+i}. Therefore, the construction of the expansion sets does not slow down the draft model's inference. It is also important to note that tokens in the expansion sets are not used for predicting future tokens. This is a key difference from proposal expansion using beam search, where each of the top-k tokens at a position is used to generate subsequent tokens, resulting in expanded proposals in tree structures with many branches. Such beam search-based expansion would be much more expensive to compute and thus less efficient, which we will demonstrate in §5.

Verification of Expanded Proposals. Because our proposals now contain additional candidate tokens at each position, the verification mechanism also needs to be modified. We borrow ideas from the token tree verifier (Miao et al., 2023; Cai et al., 2023; Sun et al., 2023) and implement our verification procedure as follows. First, we *linearize* an expanded proposal into a single sequence by simply appending the tokens in the expansion sets to the end of the proposal. We then pass this sequence to $\mathcal{M}_T$ for parallel verification with a special causal mask. The key idea of this mask is to ensure that each token, regardless of whether it is in the original proposal or in an expansion set, attends to only those tokens in the original proposal in front of this token and the token itself.

Mathematically, the mask is defined as follows. Let $\mathcal{P} = (x_{t+1}, \ldots, x_{t+\beta})$ denote the linearized expanded proposal, where the first γ tokens are from the original proposal and the remaining tokens are from the expansion sets. Let $\mathrm{pos}(i)$ be a function that maps the token x_{t+i} in $\mathcal{P}$ to the token's original position. That means, for $i > \gamma$, $\mathrm{pos}(i) = j$ where $x_{t+i} \in \mathcal{X}_j$. Let A denote a $\beta \times \beta$ matrix representing the attention scores between the tokens in $\mathcal{P}$. The mask function is defined as:

$$\mathrm{Mask}_{\mathrm{CAPE}}(A)_{ij} = \begin{cases} A_{ij} & \text{if } j \leq \gamma \text{ and } \mathrm{pos}(i) > j \\ A_{ij} & \text{if } i = j \\ -\infty & \text{otherwise} \end{cases}.$$

Here the condition $j \leq \gamma$ is to check whether x_{t+j} is a token from the original proposal, the condition $\mathrm{pos}(i) > j$ is to ensure that token x_{t+i} is after token x_j, and the condition $i = j$ is for self-attention. The pseudo-code can be found in Appendix A.

5. Experiments

5.1. Settings

Target and draft models. We select two widely-used LLMs, Vicuna (including 7b, 13b, and 33b) (Chiang et al., 2023) and Mistral (7b-instruct-v0.1) (Jiang et al., 2023), as target models. To make the draft model more efficient, we choose a wider and shallower architecture. Specifically, for 7b and 13b target models, we use set N_{D} to 1, and for the

Models	Cost	GSM8K		Fin.-Alp.		Spider		Code.	
	↓ (%)	Acc. (%)	$\mathbb{E}$(Spd.)	Acc. (%)	$\mathbb{E}$(Spd.)	Acc. (%)	$\mathbb{E}$(Spd.)	Acc. (%)	$\mathbb{E}$(Spd.)
$\mathcal{M}_T$ **Vicuna-7B**									
LLAMA-68m	7.7	51.6	1.46	49.7	1.41	19.2	0.89	34.8	1.11
$\mathcal{M}_D$ LLAMA-160m	49.0	58.0	0.66	57.1	0.65	25.1	0.38	40.0	0.48
GLIDE	**6.7**	**64.8**†	**1.97**†	**63.2**†	**1.90**†	**55.7**†	**1.64**†	**67.0**†	**2.06**†
$\mathcal{M}_T$ **Vicuna-13B**									
LLAMA-68m	6.2	50.2	1.51	49.3	1.48	19.0	0.94	32.7	1.13
$\mathcal{M}_D$ LLAMA-160m	39.3	56.8	0.75	56.8	0.75	25.4	0.45	38.3	0.54
GLIDE	**5.5**	**67.1**†	**2.16**†	**65.0**†	**2.07**†	**57.1**†	**1.77**†	**68.6**†	**2.24**†
$\mathcal{M}_T$ **Vicuna-33B**									
LLAMA-68m	**4.6**	49.1	1.57	46.5	1.50	30.0	1.16	31.5	1.19
$\mathcal{M}_D$ LLAMA-160m	29.5	56.1	0.89	53.1	0.84	31.4	0.59	36.7	0.64
GLIDE	**6.8**	**69.3**†	**2.16**†	**64.3**†	**1.94**†	**62.7**†	**1.87**†	**68.9**†	**2.14**†
$\mathcal{M}_T$ **Mistral-7B-Ins.**									
$\mathcal{M}_D$ LLAMA-45m*	**4.9**	36.7	1.27	40.2	1.34	41.9	1.38	44.5	1.44
GLIDE	**6.6**	**60.1**†	**1.80**†	**56.4**†	**1.67**†	**59.8**†	**1.79**†	**62.6**†	**1.89**†

Table 1. Comparison between GLIDE and previous draft models. † denotes results that are statistically significantly better than the corresponding best LLAMA draft model with $p < 0.01$. *Since there is no open-source draft model trained for mistral, we retrained it using the same data of GLIDE following the setting of Miao et al. (2023). The walltime speedup is listed over Table 5.

33b target model, we set N_{D} to 2. We set d_{D} to be 4096. Standard cross entropy is used to optimize the draft model while the parameters of the target model are kept frozen. More details on model training can be found in Appendix B.

Datasets. We first train our draft model on the pre-training dataset SlimPajama-6B (Soboleva et al., 2023). We then finetune the draft model on a supervised-finetuning (SFT) dataset (ShareGPT (GPT3.5 & 4, 2023) in our case) to further improve the model performance. Following Liu et al. (2023), we evaluate our GLIDE method across four different datasets: GSM8K (Cobbe et al., 2021) (math reasoning), Finance-Alpaca (Bharti, 2023) (QA for finance), Spider (Yu et al., 2018) (text-to-SQL), and Code-Search-Python (Husain et al., 2020) (Python code generation). We follow (Cai et al., 2023) and use the well-known benchmark dataset MT-Bench (Zheng et al., 2023) for the evaluation of CAPE.

Metrics. A widely-used metric for SD is *acceptance rate* α (Leviathan et al., 2023), which is the expected probability of the target model accepting a token proposed by the draft model. Leviathan et al. (2023) showed that with speculative sampling, α is equivalent to $\mathbb{E}_{x\sim p_{\mathrm{D}}(x)} \min(p_{\mathrm{T}}, p_{\mathrm{D}})$, where p_{T} and p_{D} are the target and the draft models' next token probabilities. The acceptance rate is independent of hardware configuration and therefore a more objective metric.

Leviathan et al. (2023) defined the *cost coefficient* c to be the ratio between the walltime of a single run of $\mathcal{M}_{\mathrm{D}}$ and that of $\mathcal{M}_{\mathrm{T}}$. Given proposal length γ, acceptance rate α, and cost coefficient c, Leviathan et al. (2023) derived the following formula for the expected improvement factor in total walltime, or *expected speedup*.

$$\mathbb{E}(\text{Spd.}) = \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}.$$

Finally, we also use actual decoding speed and walltime speedup for additional comparison.

Experiment design. To verify the effectiveness of GLIDE and CAPE separately, we design two sets of experiments. First, we employ only GLIDE without CAPE and use acceptance rate and expected speedup as evaluation metrics. We set proposal length γ to be 5 and adopt speculative sampling as our acceptance strategy, following (Liu et al., 2023). Next, we compare GLIDE+CAPE with Medusa (Cai et al., 2023) and GLIDE+BeamSearch.

For GLIDE+BeamSearch, during speculation, instead of generating a single proposal sequence $(x_{t+1}, x_{t+2}, \ldots, x_{t+\gamma})$, we will generate k proposal sequences, where k is the beam size. Specifically, at position $(t+1)$, we use the draft model to find the top-k tokens to form the best k proposals up to position $(t+1)$.

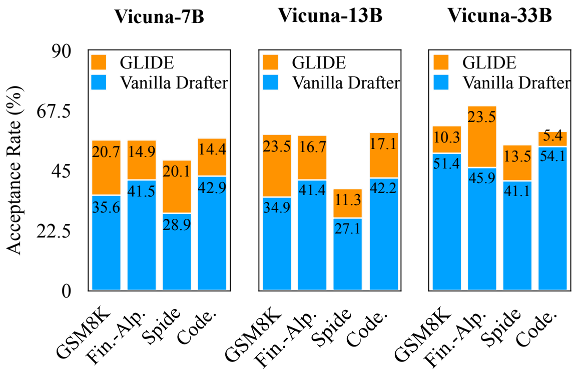


Figure 4. Comparison between GLIDE and Vanilla Drafter. The oranges boxes show the *improvement* brought by GLIDE.

Then at position $(t + 2)$, using any of the top-k proposals up to position $(t + 1)$ as the context/prefix, we use the draft model to score the next token and find the top-k proposals up to position $(t + 2)$ that have the highest overall scores. We continue this process until we reach position $(t + \gamma)$ and obtain the top-k proposal sequences of length γ. These proposal sequences can be represented by a tree structure. We adopt a variant of the tree-based parallel verification method described in SpecInfer (Miao et al., 2023) to perform efficient verification. We experimented with $k = 4$ and $k = 8$, and found $k = 4$ to have a smaller speculation time and better walltime speedup. The performance of GLIDE+BeamSearch reported in the paper is based on $k = 4$.

Both Medusa and BeamSearch generate multiple proposals and use a tree verification mechanism. For these methods that generate multiple proposals, the metric *acceptance rate* is no longer meaningful, and therefore we use walltime speedup as our main metric. While we could use either speculative sampling or speculative decoding as the acceptance strategy, to be consistent with the evaluation of Medusa (Cai et al., 2023), here we adopt speculative decoding as our acceptance strategy, and the batch size is set to be 1.

5.2. Evaluation of GLIDE

Effectiveness of attending to target model's KV cache. To verify whether it is effective to use the target model's KV cache, we first conduct a controlled experiment where we compare two draft model architectures: (1) GLIDE, which uses cross-attention to re-use the target model's KV cache, and (2) Vanilla Drafter, which has the same architecture as GLIDE except that the cross-attention sub-layers are removed. To reduce the experimental cost, we randomly sample half of the data from SlimPajma-6B to train both GLIDE and the Vanilla Drafter. We further fine-tune the two models on ShareGPT. We run the experiments on GSM8K, Fin.-Alp., Spider, and Code. We find that GLIDE always has a substantially higher acceptance rate than Vanilla Drafter. In Figure 4, we show the improvement in terms of acceptance rate by GLIDE over Vanilla Drafter on the four datasets. We can see that the improvement of acceptance rate ranges between 5.4 and 23.5 percentage points, and in most settings the improvement is over 10 percentage points. Given such substantial improvement, we can conclude that it is highly effective for the draft model to re-use the target model's KV cache to improve the quality of the proposed sequences and thus improve the acceptance rate.

Comparison between GLIDE and other draft models. Next, we compare our GLIDE draft model trained on the entire set of SlimPajma-6B and finetuned on ShareGPT with a few previous baseline draft models, namely, LLAMA-68m (Miao et al., 2023), LLAMA-160m (Miao et al., 2023), and LLAMA-45m. Comparison between our GLIDE and these baseline draft models is shown in Table 1. We show three metrics: Cost (cost coefficient), Acc. (acceptance rate), and $\mathbb{E}$(Spd.) (expected speedup).

From the table, we have the following findings. (1) The cost coefficient of GLIDE is very low, comparable to that of LLAMA-68m or LLAMA-45m. Recall that the cost coefficient measures the relative walltime cost of the draft model compared with the target model, and a 5%-6% cost coefficient means the walltime cost of our GLIDE draft model is negligible compared with that of the target model. (2) In terms of acceptance rate and expected speedup, our GLIDE draft model clearly beats the baseline draft models under all settings, and the improvement is all statistically significant. (3) The acceptance rate of GLIDE generally falls between 55% and 70%, which means more than half of the proposed tokens are accepted. In comparison, the baseline draft models sometimes only have an acceptance rate of around 30%. Similarly, the expected speedup of GLIDE ranges between 1.67 and 2.24, much higher than those of the baseline draft models, which is always below 1.5 for LLAMA-68m and LLAMA-45m and below 1.0 for LLAMA-160m. In sum, the results in Table 1 again demonstrate that GLIDE is highly effective.

Comparison of actual decoding speed. To see the actual speedup of GLIDE, we also report the actual decoding speed in terms of number of tokens per second based on MT-bench (Zheng et al., 2023). The comparison between the speed of GLIDE and that of the baselines is shown in Figure 5. We find that GLIDE substantially accelerates the model's decoding speed. The results are consistent with the findings with the expected speedup. It is interesting to note that when using GLIDE, our accelerated vicuna-33b model has a faster decoding speed than vicuna-7b without speculative decoding.

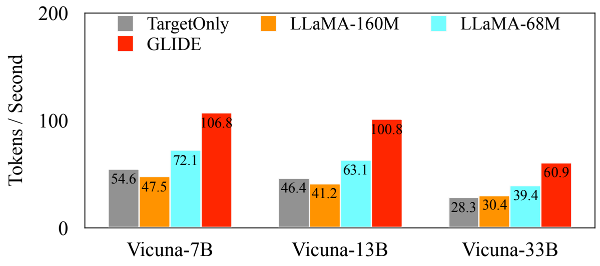


Figure 5. The decoding speed (tokens per second) on MT-Bench.

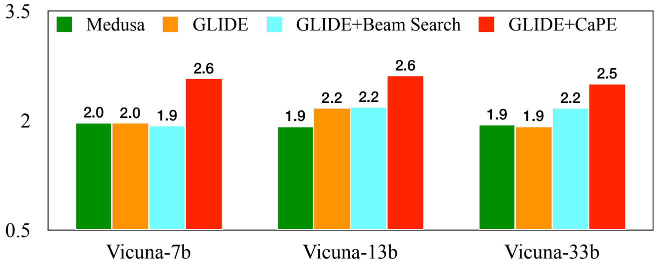


Figure 6. The practical speedup with GLIDE + CAPE.

5.3. Evaluation of CAPE

Next, we assess the effectiveness of CAPE on top of GLIDE. It is worth noting that acceptance rate based on its original definition is not applicable to the setting when the draft model proposes additional candidate tokens for verification. Therefore, in this set of experiments, we directly measure the walltime speedup on MT-Bench, in line with Medusa. Here, the walltime speedup is defined to be the ratio between the walltime of applying the speculative decoding method under evaluation and the walltime of standard decoding using only the target model. We compare GLIDE+CAPE with the following baselines: (1) Medusa (Cai et al., 2023), which uses a non-autoregressive draft model to parallel generate multiple proposals. (2) GLIDE, which does not use CAPE. (3) GLIDE+BeamSearch, where we set the beam size equal to 4. Medusa and GLIDE+BeamSearch use a tree attention mechanism (Miao et al., 2023; Cai et al., 2023) for efficient verification. The results are shown in Figure 6.

We have the following findings from the figure. (1) GLIDE+CAPE clearly outperforms all the baselines with a substantial margin. (2) There is a clear improvement of walltime speedup when CAPE is added on top of GLIDE. This illustrates the usefulness of further employing CAPE on top of GLIDE. (3) It is interesting to observe that even without CAPE, GLIDE alone performs better than Medusa. This again shows the effectiveness of GLIDE itself. It is worth pointing out that Medusa also uses the hidden states from the target model for its draft model's proposal generation, but instead of using all the previous KV cache like what we do, Medusa uses the the hidden state of the last verified tokens only, which may affect the quality of the draft model's predictions. Furthermore, Medusa uses a non-autoregressive way to generate proposal sequences. It is well known that non-autoregressive language models tend to generate output sequences with lower fluency (Gu et al., 2018; Du et al., 2021; 2022). In comparison, our GLIDE model is an autoregressive model. Therefore, the quality of the proposals by Medusa is likely lower than ours. We suspect that this is another important factor for Medusa to perform worse than our method in terms of walltime speedup. (4) Another interesting finding is that GLIDE+BeamSearch may slower than GLIDE. This shows that simply employing beam search to generate multiple proposal sequences is not guaranteed to work. Although the additional proposal sequences may increase the chance of acceptance, computationally, generating these proposals during speculation and verifying them during verification incur additional costs, which in our experiments seem to outweigh their benefits. We conduct further analysis in §5.4.

It is worth noting that our GLIDE+CAPE method is not only faster than Medusa but also verifies a smaller number of proposed tokens in each batch than Medusa. Our CAPE sets the maximum number of tokens for verification at each step to be 32, whereas Medusa's is 64. So our CAPE may support larger batch size inference.

5.4. Further Analysis

Impact of KV cache at different layers. In our default setting, the draft model attends to the KV cache from the top layers of the target model, as described in §4. It is also possible to attend to lower layers' KV cache. To see whether indeed KV cache from higher layers is more effective, we compare the acceptance rates when GLIDE is used with Mistral-7b as the target model on the four datasets. The results are shown in Figure 7. We can see that clearly using KV cache from higher layers of the target model produces higher acceptance rates, confirming our assumption that using the KV cache from the top layers is more effective. On the other hand, using KV cache from lower layers is still useful compared with not using KV cache at all. Therefore, GLIDE can potentially be combined with early exit speculative decoding methods (Schuster et al., 2022) to further reduce the overall decoding time.

CAPE vs. beam search. Earlier in Figure 6 we find that while CAPE reduces walltime, beam search incurs more walltime. To better understand the issue, we separately compare the speculation time and verification time. We find that for verification, CAPE and beam search have similar walltime. But for each step of speculation, beam search increases the walltime from 5.5ms to 10.9ms (a difference of 5.4ms) on average, whereas CAPE only increases the walltime by 0.2ms. The main reason for beam search to increase walltime so much is that beam search requires copying and sorting operations that are very time-consuming. This phe-

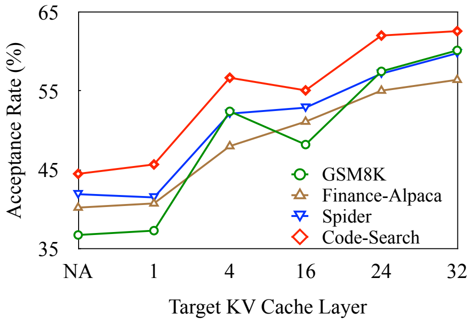


Figure 7. The relationship between the acceptance rate and the n-th layer of target model for KV cache.

nomenon was also observed by other researchers[1].

The importance of confidence scores in CAPE. Recall that CAPE uses the draft model's confidence scores to dynamically determine the number of additional tokens in each expansion set. To see whether this confidence-aware expansion is useful, we compare CAPE with another proposal expansion method, where we set the size of the expansion sets to 4, which is the average size of the practical expansion sets. We find that using this fixed-size expansion method, the decoding speed decreases from 70.6 tokens/sec to 68.4 tokens/sec on MT-Bench. This shows that it is useful to use the confidence scores to adjust the sizes of different expansion sets.

6. Conclusions

In this work, we propose a draft model architecture called GLIDE that leverages the KV cache from the target model to improve its proposal generation. We also propose a confidence-aware proposal expansion mechanism called CAPE that produces additional candidate tokens for verification. Experiments demonstrate that both GLIDE and CAPE are highly effective ways to accelerate speculative decoding. Our method also substantially outperforms the strong baseline Medusa based on walltime. Overall, the integration of GLIDE with CAPE results in a 2.5x speedup on Vicuna models. As future work, we will explore the batch serving of GLIDE and its effectiveness in processing extremely long contexts and multimodal domain (Wang et al., 2019a;b; Wu et al., 2024).

Impact Statement

Our work significantly improves the inference speed at which large language models make predictions, thus helping to make AI technologies more accessible and opening up more possibilities for personal AI applications. However, since our method does not change the output generated by LLMs, it means our approach could inadvertently speed up the creation of harmful or biased content, such as hate speech or misinformation. This highlights the critical need for careful use and the establishment of strong safeguarding measures to reduce the dangers linked to enhanced LLM processing speeds.

Acknowledgement

Cunxiao wants to especially thank Xuan, Fangkai, and Jianlin's help with this paper. Yang You is being sponsored by NUS startup grant (Presidential Young Professorship), Singapore MOE Tier-1 grant, ByteDance grant, ARCTIC grant, SMI grant (WBS number: A-8001104-00-00), Alibaba grant, and Google grant for TPU usage.

References

Bharti, G. Finance-alpaca dataset, 2023. URL https://huggingface.co/datasets/gbharti/finance-alpaca.

Cai, T., Li, Y., Geng, Z., Peng, H., and Dao, T. Medusa: Simple framework for accelerating llm generation with multiple decoding heads, 2023. URL https://github.com/FasterDecoding/Medusa. GitHub repository.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. M. Accelerating large language model decoding with speculative sampling. *arXiv:2302.01318*, 2023.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. URL https://vicuna.lmsys.org.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Du, C., Tu, Z., and Jiang, J. Order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Du, C., Tu, Z., Wang, L., and Jiang, J. ngram-OAXE: Phrase-based order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, 2022.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Breaking the sequential dependency of llm inference using looka-

---

[1] https://github.com/ggerganov/llama.cpp/issues/3137

head decoding, 2023. URL https://github.com/hao-ai-lab/LookaheadDecoding. GitHub repository.

GPT3.5 and 4, G. Sharegpt dataset, 2023. URL https://github.com/domeccleston/sharegpt.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. Non-autoregressive neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

He, Z., Zhong, Z., Cai, T., Lee, J. D., and He, D. Rest: Retrieval-based speculative decoding, 2023.

Husain, H., Wu, H.-H., Gazit, T., Allamanis, M., and Brockschmidt, M. Codesearchnet challenge: Evaluating the state of semantic code search. 2020.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *arXiv:2310.06825*, 2023.

Jiao, W., Wang, W., Huang, J.-t., Wang, X., and Tu, Z. Is chatgpt a good translator? yes with gpt-4 as the engine. *arXiv:2301.08745*, 2023.

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Speculative decoding with big little decoder. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Liu, X., Hu, L., Bailis, P., Stoica, I., Deng, Z., Cheung, A., and Zhang, H. Online speculative decoding. *arXiv:2310.07177*, 2023.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. Specinfer: Accelerating generative large language model serving with speculative inference and token tree verification. *arXiv:2305.09781*, 2023.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., and Metzler, D. Confident adaptive language modeling. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL https://huggingface.co/datasets/DKYoon/SlimPajama-6B.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Sun, Z., Suresh, A. T., Ro, J. H., Beirami, A., Jain, H., Yu, F., Riley, M., and Kumar, S. Spectr: Fast speculative decoding via optimal transport. In *Workshop on Efficient Systems for Foundation Models @ ICML*, 2023.

Wang, X., Wu, J., Chen, J., Li, L., Wang, Y.-F., and Wang, W. Y. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4581–4591, 2019a.

Wang, X., Wu, J., Zhang, D., Su, Y., and Wang, W. Y. Learning to compose topic-aware mixture of experts for zero-shot video captioning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 8965–8972, 2019b.

Wu, X., Yu, S., Lim, E.-P., and Ngo, C.-W. Ovfoodseg: Elevating open-vocabulary food image segmentation via image-informed textual representation. *arXiv preprint arXiv:2404.01409*, 2024.

Xia, H., Ge, T., Wang, P., Chen, S.-Q., Wei, F., and Sui, Z. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2024.

Yang, N., Ge, T., Wang, L., Jiao, B., Jiang, D., Yang, L., Majumder, R., and Wei, F. Inference with reference: Lossless acceleration of large language models. *arXiv:2304.04487*, 2023.

Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for {Transformer-Based} generative models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022a.

Yu, S., Sun, Q., Zhang, H., and Jiang, J. Translate-train embracing translationese artifacts. Association for Computational Linguistics, 2022b.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Yuan, H., Lu, K., Huang, F., Yuan, Z., and Zhou, C. Speculative contrastive decoding. *arXiv:2311.08981*, 2023.

Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft verify: Lossless large language model acceleration via self-speculative decoding. *arXiv:2309.08168*, 2023.

Zhang, X. and Gao, W. Towards LLM-based fact verification on news claims with a hierarchical step-by-step prompting method. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 996–1011, 2023.

Zhang, X. and Gao, W. Reinforcement retrieval leveraging fine-grained feedback for fact checking news claims with black-box llm. *arXiv preprint arXiv:2404.17283*, 2024.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS Datasets and Benchmarks Track*, 2023.

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. Distillspec: Improving speculative decoding via knowledge distillation. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

A. Pseudo Code of CAPE

See Algorithm 1 for the proposal expansion and Algorithm 2 for the verification.

Algorithm 1 Confidence-Aware Proposal Expansion.

```
Definitions
$\mathcal{M}_D$ - draft model
$S = [8, 8, 8, 6, 6, 6, 4, 4, 2, 2]$
    - confidence bin array
$P$ - probability distribution for each step
$R$ - proposal results
$\mathcal{X}_i$ - extension set for the $i$-th decoding step
$T$ - greedy search result


▷ Execute Draft Model
for $i \in$ range(proposal steps) do
    $p_i \leftarrow \mathcal{M}_D(History, t_{<i})$
    $t_i \leftarrow$ argmax($p_i$)
    $T$.append($t_i$)
    $P$.append($p_i$)
end for


▷ Proposal Expansion
for $i \in$ range(proposal steps) do
    $p_i \leftarrow P[i]$
    expand_num $\leftarrow S$ [floor(10 $\times$ max($p_i$))]
    $K_i \leftarrow$ topk($p_i$, expand_num).indices
    $K$.append($K_i$)
end for


▷ Concatenation of Expanded Results
$R$.append($T$)
for $i \in$ range(proposal steps) do
    for $t \in \mathcal{X}_i$ do
        $r \leftarrow$ concat($T[0 : i], t$)
        $R$.append($r$)
    end for
end for

return $R$
```

Algorithm 2 Verification Input for $\mathcal{M}_T$.

```
Input: Proposal tokens set by cape $R$
Output: Input sequence, Input position id, Verification Attention Mask ($VAM$)

Definitions:
    $T$ - greedy search result $R[0]$
    $\mathcal{X}_i$ - extension set for the $i$-th decoding step
    $input$ - input sequence
    $input\_pi$ - input position id
    $CAM$ - Causal Attention Mask
    $VAM$ - Verification Attention Mask
    $pos$ - function that maps a position to decoding step

$\gamma \leftarrow$ len($T$)
$mask\_length \leftarrow \gamma$
$input \leftarrow T$
$input\_pi \leftarrow$ list(range($0, \gamma$))

for $i \in$ range($\gamma$) do
    for $t \in \mathcal{X}_i$ do
        $input \leftarrow$ concat($input, t$)
        $input\_pi \leftarrow$ concat($input\_pi, i$)
    end for
    $mask\_length \leftarrow mask\_length +$ len($\mathcal{X}_i$)
end for

$CAM \leftarrow$ causal_mask($mask\_length, mask\_length$)
$VAM \leftarrow$ identity($mask\_length, mask\_length$)
$VAM[: \gamma, : \gamma] \leftarrow CAM[: \gamma, : \gamma]$

for $i \in$ range($\gamma, mask\_length$) do
    for $j \in$ range($\gamma$) do
        if $pos(i) > j$ then
            $VAM[i][j] \leftarrow CAM[i][j]$
        end if
    end for
end for

return $input$, $input\_pi$, $VAM$
```

Models	GSM8K	Fin.-Alp.	Spider	Code-Py.
w/o Dist.	64.8	63.2	55.7	67.0
w Dist.	65.0	63.9	56.0	67.5

Table 2. Impact of Distillation on GLIDE w.r.t acceptance rate (%).

Models	GLIDE	Beam 4	Beam 8	+CAPE
Vicuna-7B	5.3	10.9	14.8	5.5
Vicuna-13B	5.3	10.9	14.9	5.5
Vicuna-33B	9.1	17.1	24.6	8.9

Table 3. The speed of draft model's speculation (ms).

B. Experiment Details

For the 7B and 13B target language models, we employ a single-layer GLIDE with a hidden dimension of 4096. For the 33B target model, we use a two-layer GLIDE also with a hidden dimension of 4096.

In the case of the 7B and 13B target models, we train GLIDE with zero2 and eight H800 GPUs. For the 33B target model, we use zero3 and 16 H800 GPUs. We set batch size (with accumulation) as 64, learning rate equals to 5e-4, and use adamW (Kingma & Ba, 2015) to optimize the draft model. We only train our draft model for one epoch on both pretrain and SFT datasets.

For the 7B and 13B models, the training is approximately 7 hours and 10 hours, respectively, whereas for the 33B target model, it takes about 100 hours.

As discussed in the section limitations, the main training expense lies in forwarding the target model to get the KV cache. If we could incorporate GLIDE during the training of the LLM, this time-consuming part could be omitted. All the inference processes in this paper are performed using fp16 and on a single H800 GPU.

C. Impact of Distillation

Distillation is a common technology for speculative decoding (Zhou et al., 2024; Liu et al., 2023). We use seq-level distillation (Kim & Rush, 2016) for GLIDE to see the benefit. Table 2 shows that our GLIDE can be further improved via distillation. However, the training time cost of seq-level distillation is very expensive, which is 2 times than the training time of vanilla GLIDE. So we do not use distillation in other experiments.

D. Walltime for Speculation

We provide more walltime results at Table 3, it can be observed that with larger beam, the speculation walltime becoming longer.

E. Batch Serving

Here we also show the decoding speed of our proposed GLIDE at Table 4. Please note we do not design any specific algorithm for batch serving, so the experiment here is similar to the vanilla draft model. To avoid other irrelevant reasons like different samples in the same batch having different lengths, we just duplicate the input IDs to batch numbers. We test the performance using one H800 and fp16 as the platform and setting.

As Table 4 shows, if batch sizes are not large enough, it will not largely degenerate the decoding speed. That is because, under the setting of small batch size, the inference process of LLM is memory-bounded. However, we observe that OOM frequently occurs during for the long prompts, even for the target model only. So we believe how to combine speculative decoding and batch serving systems (e.g., vLLM (Kwon et al., 2023), continuous batching (Yu et al., 2022a)) will be an important topic in the future.

F. Configurations for GLIDE

In our preliminary exploratory research, we discover that the structure of draft models has a significant impact on the speedup ratio. We find that even for the 12-layers llama-160m could lead to a decrease in decoding speed. This is because there is a clear sequential order between layers, necessitating autoregressive operation. Although the multi-layered draft model can significantly improve acceptance rates, this tradeoff is unwise.

We start with the structure of llama-68m, i.e., 2 hidden layers + hidden dimension=768, and do a grid search to find a more reasonable architecture. First, we try settings of 1, 2, and 4 hidden layers (for Vicuna 7B, we only experimented with 1 and 2 layers). We find that for the target models smaller than 33B, the optimal architecture is a single-layer draft model, whereas for the 33B LLM, the optimal architecture is two layers as shown in Fig. 8. Next, we conduct a grid search on the hidden size. We conduct three sets of experiments on the 7B and 13B models, with hidden dimensions of 768, 2048, and 4096, respectively as shown in Figure 9. We find that increasing the hidden size can effectively compensate for the loss caused by the reduction of hidden layers, and does not significantly increase operational speed.

G. Walltime Speedup for Main the Experiment

We list the wall time speedup for the datasets used in main experiments at Table 5.

Batch Size	Vicuna-7B		Vicuna-13B		Vicuna-33B	
	+GLIDE	Only	+GLIDE	Only	+GLIDE	Only
1	106.8	54.6	100.8	46.2	60.9	28.3
2	2 * 108.2	2 * 55.6	2 * 99.5	2 * 44.5	2 * 59.8	2 * 27.9
4	4 * 116.5	4 * 55.9	4 * 100.8	4 * 42.9	OOM	OOM
8	8 * 108.5	8 * 55.1	8 * 98.5	8 * 43.2	OOM	OOM
16	16 * 106.2	16 * 55.4	OOM	16 * 43.2	OOM	OOM
32	OOM	OOM	OOM	OOM	OOM	OOM

Table 4. The speed (tokens/sec) of GLIDE under different batch sizes settings. OOM denotes out of the HBM memory of 80GB H800.

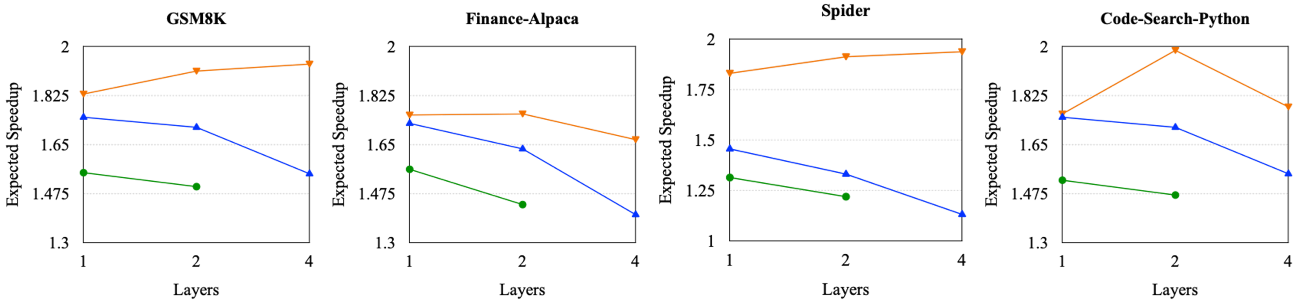


Figure 8. The relationship between theoretical speedup and the number of layers in a draft model is such that when the target LLM has fewer layers, increasing the number of layers in the draft model results in a significant decrease in performance. This is because, compared to the improvement in the acceptance rate brought about by adding layers to the draft model, the increase in time cost is greater.

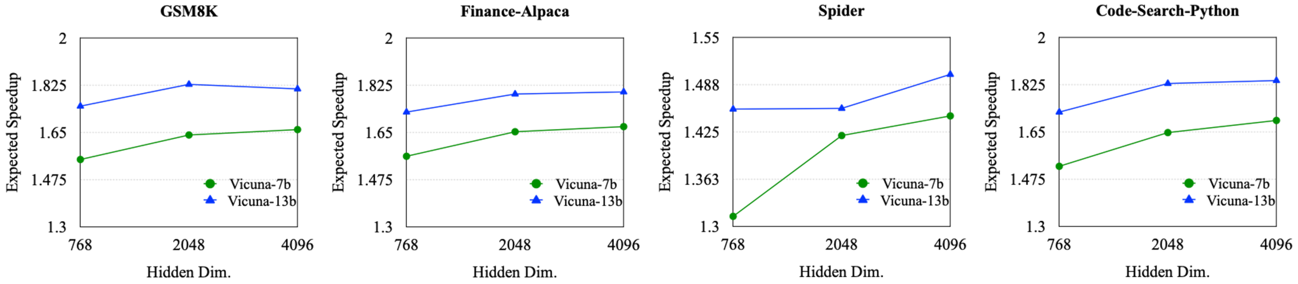


Figure 9. The relationship between theoretical speedup and the hidden size of the draft model is such that increasing the hidden size can significantly accelerate the process. This is because the elements within the hidden layer can be computed in parallel, whereas the computation between layers must be done in an autoregressive manner.

Target	Draft	GSM.	Fin.	Spider	Code
Vic. 7B	Llama-68m	1.53	1.48	0.95	1.02
Vic. 7B	**GliDe Only**	**1.90**	**1.94**	**1.69**	**2.11**
Vic. 13B	Llama-68m	1.54	1.57	0.94	1.17
Vic. 13B	**GliDe Only**	**2.32**	**2.07**	**1.82**	**2.35**
Vic. 33B	Llama-68m	1.56	1.47	1.07	1.19
Vic. 33B	**GliDe Only**	**2.27**	**2.02**	**1.95**	**2.14**

H. Limitation

Although our model significantly improves acceptance rates, it does have some inherent limitations. The most notable issue is that our model is not a plug-and-play algorithm. This is primarily because our draft model relies on reusing the KV cache of the LLM, necessitating the retraining of a corresponding draft model for speculative decoding each time a new LLM is introduced. However, considering the relatively low cost of this process, which is approximately 7 hours for training a 68MB draft model on 8 H800 GPU cards, this limitation can be considered manageable within practical applications. We must also note that for vanilla speculative decoding when the tokenizer of the LLM (e.g., Mistral vs Vicuna) is changed, it is necessary to retrain a draft model that is compatible with the new LLM. In the training of GLIDE, the bottleneck is the forward of the target LLM instead of the lightweight draft model. Therefore, we suggest that future open-source models can be trained simultaneously with a compatible GLIDE, thus avoiding the time-consuming re-forward target LLM.

I. Case Study

Table 6 presents an illustrative case from MT-bench, where tokens marked in red indicate acceptance. The vanilla draft model exhibits a notable deficiency in acceptance rate, evident through a scarcity of red tokens. In contrast, our proposed GLIDE demonstrates a markedly higher acceptance of tokens, contributing to increased decoding speed. Notably, GLIDE effectively predicts terms such as "bacteria" and "germs", due to its enhanced drafting capabilities. Furthermore, the integration of GLIDE with CAPE shows further improvement in token acceptance, corresponding to the experimental results in Figure 5. More cases can be found at the following.

Table 6. Random Sampled example from MT-bench, red tokens denotes accepted tokens. The target model is Mistral-7B-instruct-v0.1.

User Input	Embrace the role of Sheldon from "The Big Bang Theory" as we delve into our conversation. Don't start with phrases like "As Sheldon". Let's kick things off with the following question: "What is your opinion on hand dryers?"
LLAMA-45m	Hand dryers are a convenient and efficient way to dry hands after using the restroom or washing them. They can help reduce the spread of germs and bacteria, as well as save time and paper towels. However, some people may prefer using paper towels for personal reasons or because they feel more comfortable with the sensation. Ultimately, it comes down to personal preference and what works best for each individual.
GLIDE	Hand dryers are a convenient and efficient way to dry hands after using the restroom or washing them. They can help reduce the spread of germs and bacteria, as well as save time and paper towels. However, some people may prefer using paper towels for personal reasons or because they feel more comfortable with the sensation. Ultimately, it comes down to personal preference and what works best for each individual.
+ CAPE	Hand dryers are a convenient and efficient way to dry hands after using the restroom or washing them. They can help reduce the spread of germs and bacteria, as well as save time and paper towels. However, some people may prefer using paper towels for personal reasons or because they feel more comfortable with the sensation. Ultimately, it comes down to personal preference and what works best for each individual.

Table 7. Random Sampled example from MT-bench, red tokens denotes accepted tokens. Target model is Vicuna-7B-1.5.

User Input	Write a descriptive paragraph about a bustling marketplace, incorporating sensory details such as smells, sounds, and visual elements to create an immersive experience for the reader.
LLAMA-45m	As I walk through the bustling marketplace, I am immediately struck by the sights and sounds of the vibrant atmosphere. The air is thick with the scent of exotic spices and freshly baked goods, mingling with the sounds of vendors calling out to customers and the clatter of plates and bowls being set out on display. The marketplace is a riot of colors and textures, with stalls selling everything from brightly hued fabrics to intricately carved wooden sculptures. The ground is covered in a thick layer of colorful pebbles, and the walls are adorned with vibrant murals that seem to come alive as I walk by. The marketplace is a sensory feast, and I can't help but feel a sense of excitement and wonder as I take it all in.
GLIDE	As I walk through the bustling marketplace, I am immediately struck by the sights and sounds of the vibrant atmosphere. The air is thick with the scent of exotic spices and freshly baked goods, mingling with the sounds of vendors calling out to customers and the clatter of plates and bowls being set out on display. The marketplace is a riot of colors and textures, with stalls selling everything from brightly hued fabrics to intricately carved wooden sculptures. The ground is covered in a thick layer of colorful pebbles, and the walls are adorned with vibrant murals that seem to come alive as I walk by. The marketplace is a sensory feast, and I can't help but feel a sense of excitement and wonder as I take it all in.
+ CAPE	As I walk through the bustling marketplace, I am immediately struck by the sights and sounds of the vibrant atmosphere. The air is thick with the scent of exotic spices and freshly baked goods, mingling with the sounds of vendors calling out to customers and the clatter of plates and bowls being set out on display. The marketplace is a riot of colors and textures, with stalls selling everything from brightly hued fabrics to intricately carved wooden sculptures. The ground is covered in a thick layer of colorful pebbles, and the walls are adorned with vibrant murals that seem to come alive as I walk by. The marketplace is a sensory feast, and I can't help but feel a sense of excitement and wonder as I take it all in.

Table 8. Random Sampled example from MT-bench, red tokens denotes accepted tokens. Target model is Vicuna-7B-1.5.

User Input	Picture yourself as a 100-years-old tree in a lush forest, minding your own business, when suddenly, a bunch of deforesters shows up to chop you down. How do you feel when those guys start hacking away at you?
LLAMA-45m	As a 100-years-old tree, I have seen many changes in the world around me. I have witnessed the growth of forests and the decline of forests. I have seen the rise and fall of civilizations. I have seen the impact of human activity on the environment. When the deforesters show up and start hacking away at me, I feel a sense of loss and sadness. I have lived a long and fulfilling life, and now my time has come to an end. I am not afraid of death, but I am sad to leave this world behind. I also feel a sense of anger and frustration towards the deforesters. They are destroying the very thing that they need to survive - the forests that provide oxygen, clean water, and habitat for countless species. They are acting recklessly and irresponsibly, without considering the long-term consequences of their actions. Despite my feelings, I know that I cannot stop them. I am just a tree, after all. But I hope that my sacrifice will serve as a reminder to future generations of the importance of protecting the environment and preserving the natural world for future generations to enjoy.
GLIDE	As a 100-years-old tree, I have seen many changes in the world around me. I have witnessed the growth of forests and the decline of forests. I have seen the rise and fall of civilizations. I have seen the impact of human activity on the environment. When the deforesters show up and start hacking away at me, I feel a sense of loss and sadness. I have lived a long and fulfilling life, and now my time has come to an end. I am not afraid of death, but I am sad to leave this world behind. I also feel a sense of anger and frustration towards the deforesters. They are destroying the very thing that they need to survive - the forests that provide oxygen, clean water, and habitat for countless species. They are acting recklessly and irresponsibly, without considering the long-term consequences of their actions. Despite my feelings, I know that I cannot stop them. I am just a tree, after all. But I hope that my sacrifice will serve as a reminder to future generations of the importance of protecting the environment and preserving the natural world for future generations to enjoy.
+ CAPE	As a 100-years-old tree, I have seen many changes in the world around me. I have witnessed the growth of forests and the decline of forests. I have seen the rise and fall of civilizations. I have seen the impact of human activity on the environment. When the deforesters show up and start hacking away at me, I feel a sense of loss and sadness. I have lived a long and fulfilling life, and now my time has come to an end. I am not afraid of death, but I am sad to leave this world behind. I also feel a sense of anger and frustration towards the deforesters. They are destroying the very thing that they need to survive - the forests that provide oxygen, clean water, and habitat for countless species. They are acting recklessly and irresponsibly, without considering the long-term consequences of their actions. Despite my feelings, I know that I cannot stop them. I am just a tree, after all. But I hope that my sacrifice will serve as a reminder to future generations of the importance of protecting the environment and preserving the natural world for future generations to enjoy.