MIXTURE OF ATTENTIONS FOR SPECULATIVE DECODING

Matthieu Zimmer[*], **Milan Gritta**[*] **& Gerasimos Lampouras**
Huawei Noah's Ark Lab, firstname.lastname@huawei.com

Haitham Bou Ammar[‡]
Huawei Noah's Ark Lab, University College London

Jun Wang[‡]
University College London

ABSTRACT

The growth in the number of parameters of Large Language Models (LLMs) has led to a significant surge in computational requirements, making them challenging and costly to deploy. Speculative decoding (SD) leverages smaller models to efficiently propose future tokens, which are then verified by the LLM in parallel. Small models that utilise activations from the LLM currently achieve the fastest decoding speeds. However, we identify several limitations of SD models including the lack of on-policyness during training and partial observability. To address these shortcomings, we propose a more grounded architecture for small models by introducing a Mixture of Attentions for SD. Our novel architecture can be applied in two scenarios: a conventional single device deployment and a novel client-server deployment where the small model is hosted on a consumer device and the LLM on a server. In a single-device scenario, we demonstrate state-of-the-art speedups improving EAGLE-2 by 9.5% and its acceptance length by 25%. In a client-server setting, our experiments demonstrate: 1) state-of-the-art latencies with minimal calls to the server for different network conditions, and 2) in the event of a complete disconnection, our approach can maintain higher accuracy compared to other SD methods and demonstrates advantages over API calls to LLMs, which would otherwise be unable to continue the generation process.

1 INTRODUCTION

Auto-regressive inference with LLMs has become quite cost-prohibitive due to the increasing parameter count of recent transformer-based LLMs (Vaswani, 2017). Different types of (usually orthogonal) solutions have been proposed to address this challenge, e.g. Mixture of Experts (Jacobs et al., 1991), Flash Attention (Dao et al., 2022), Model Quantization and Distillation (Polino et al., 2018), Linear/Sparse Self-Attention (Zhang et al., 2021), Tensor Parallelism (Shoeybi et al., 2019) and others. In this work, we focus on a recent LLM acceleration technique called Speculative Decoding, which leverages efficient models (smaller but less capable) to draft future tokens, which are verified by the LLM (more capable but much less efficient) in parallel (Leviathan et al., 2023).

The most recent state-of-the-art SD methods, like EAGLE (Li et al., 2024b) and MEDUSA (Cai et al., 2024a), leverage activations from the LLM. However, those methods have some architectural limitations including partial observability and the lack of on-policyness. Partial observability occurs when the small (draft) model lacks complete information about the state of the LLM, leading to suboptimal predictions. The lack of on-policyness during training arises because the small model is often trained under ideal conditions, assuming perfect inputs. This does not reflect the real-world scenario where the small model generates some inputs. The longer we draft new tokens using only the small model, the bigger the distribution shift from the training setting. These limitations can degrade the performance and reliability of speculative decoding.

[*] These authors contributed equally to this work

[‡] Corresponding authors: haitham.ammar@huawei.com, jun.wang@cs.ucl.ac.uk

To address these challenges, we propose a novel architecture for speculative decoding that enhances the small model's ability to accurately predict future tokens and aligns its training more closely with the inference process. Our architecture introduces several key improvements, including Layer Self-Attention (LSA) to mitigate partial observability, Cross-Attention (CA) to improve on-policyness and training efficiency, and a flexible Target Layer Inference (TLI) mechanism to balance computational efficiency and prediction accuracy. We evaluate our approach in the standard single-device setting where we demonstrate state-of-the-art speedups.

Furthermore, the SD paradigm is also ideal in the following scenario a) the model size is limited by some external factor e.g. the computational capabilities of a client device, and b) we can assume access to a larger model e.g. an LLM hosted on a server. Under this paradigm, the goal is to minimise server-side inference as well as to maintain high accuracy in the event of a total disconnection. This is an important consideration because it could pave a way for serving LLMs on edge devices, enabling them to generate responses offline while leveraging the capabilities of the large model. To this end, we extend our methodology to a client-server scenario. In this setting, we demonstrate state-of-the-art latency and minimal server calls under various network conditions (4G, 5G). Our method maintains a higher accuracy in the event of a disconnection, making it a preferred choice over independent small models or API calls that would be unable to continue generation.

Contributions We introduce a Mixture of Attentions architecture for Speculative Decoding that addresses current limitations such as partial observability as well as enabling efficient (more on-policy) training while being auto-regressive. We reuse additional activations from the LLM in the small model, enabling a trade-off between drafting speed and response quality. We conduct extensive experiments to demonstrate the effectiveness of our approach. Compared to EAGLE-2, we show a 9.5% decoding speedup with a 25% higher acceptance rate in a single-device scenario and a 84% speedup with a 53% higher acceptance rate in a client-server scenario. Finally, we propose a new framework for LLM serving in speculative client-server settings and show its effectiveness.

2 Background

We first present the background knowledge required for the remainder of the paper, i.e. the decoding mechanisms of LLMs as well as the drafting + verification techniques that ensure correct generation.

2.1 LLM Decoding

Decoding refers to the process by which LLMs generate tokens in response to input queries. This generation is typically done auto-regressively, where each new token y_t is sampled from the LLM's distribution, conditioned on both the query and the preceding tokens $y_{<t}$. We explore decoding from the perspective of dynamic systems, providing a foundation for developing new decoding mechanisms that combine large and small models (Kong et al., 2024). The internal workings of LLMs can be best understood from a dynamic system perspective, which evolves as tokens are generated. Given a large model $\mathcal{M}_{\text{Large}}$, we can describe the state transition model of vanilla decoding as:

$$\boldsymbol{h}_{\leq t+1}, \boldsymbol{o}_{t+1} = f_{\text{Large}}(\boldsymbol{h}_{\leq t}, \textit{token\_embed}(y_t)), \qquad y_{t+1} \sim \text{Softmax}(\textit{LM\_head}(\boldsymbol{o}_{t+1})), \tag{1}$$

where $\boldsymbol{h}_{\leq t}$ represents the key and value tensors of every layer until the current time-step t, y_t is the most recent token and y_{t+1} is the next token, which is sampled from a softmax distribution. Furthermore, *token_embed* is a lookup table, it assigns an embedding to a particular token of the vocabulary $\mathcal{V}$. *LM_head* is a projection from the embedding size to the vocabulary size $|\mathcal{V}|$. Finally, $f_{\text{Large}}(\cdot)$ is the function aggregating all decoder layers of $\mathcal{M}_{\text{Large}}$ and $\boldsymbol{o}_{t+1}$ is the activation of the final decoder layer. With this, the state of the dynamic system is composed of $(\boldsymbol{h}_{\leq t}, y_t)$, the minimal information needed to sample the next token from $\mathcal{M}_{\text{Large}}$.

2.2 Speculative Decoding

Some of the earliest work on speculative decoding was introduced by Stern et al. (2018), later extended to non-greedy sampling (Leviathan et al., 2023). These methods are motivated by the prohibitive cost of auto-regressive generation with $\mathcal{M}_{\text{Large}}$ that could be alleviated by using a draft model $\mathcal{M}_{\text{Small}}$ that can more efficiently generate tokens that do not require the full capability of

$\mathcal{M}_{\text{Large}}$. These hypotheses, commonly referred to as drafts, can then be verified in parallel with $\mathcal{M}_{\text{Large}}$ using rejection sampling (Leviathan et al., 2023), i.e. discard all tokens after the first mismatch. We follow the standard draft and verification cycle that iterates over the following two steps until the "end-of-sequence" token or the maximum sequence length has been reached.

1. $\mathcal{M}_{\text{Small}}$ generates new tokens $y_{t+1}, \cdots, y_{t+K}$ where K is the length of the draft sequence, auto-regressively (Xia et al., 2023; Li et al., 2024b) or in parallel (Ankner et al., 2024). The number of tokens K is typically fixed, which was the standard approach until recently when dynamic K was proposed (Nair et al., 2024; Mamou et al., 2024; Huang et al., 2024).
2. Verify K drafted tokens in a single forward pass with $\mathcal{M}_{\text{Large}}$. Verification uses either greedy (exact) matching (Xia et al., 2023), speculative sampling (Zhou et al., 2023; Leviathan et al., 2023) or 'approximate' verification (Stern et al., 2018) which relaxes the acceptance criteria but does not guarantee $\mathcal{M}_{\text{Large}}$'s output distribution. In all cases, after the first rejection, all subsequent/future tokens are typically discarded.

When the drafting of $y_{t+1}, \cdots, y_{t+K}$ is done auto-regressively, token by token, we refer to this strategy as *chain drafting*. Several works (Cai et al., 2024b; Li et al., 2024b) extended this method to *tree drafting* for additional optimisation. In this case, multiple tokens y_{t+i} can be proposed for every future position i. Verification is performed with Tree Attention (Miao et al., 2024) to efficiently handle multiple branching paths proposed by tree drafting. Consequently, this leads to an increase in acceptance lengths and reduces the number of calls to $\mathcal{M}_{\text{Large}}$ compared to chain drafting. For small batch sizes, the LLM generation is memory-bound, this is where speculative decoding can better leverage the spare compute especially with Tree Attention.

In this paper, we employ tree drafting from EAGLE-2 (Li et al., 2024b) to construct trees with a variable structure. Starting from the root node, we expand the B most probable tokens from the model $\mathcal{M}_{\text{Small}}$ $(y_{t+1}|\cdot)$. Then, for a fixed depth D, we recursively perform the following steps: for each existing branch, we compute the joint probability $\prod_{t\in D}$ $\mathcal{M}_{\text{Small}}(y_t|y_{t-1}, \cdots)$ of their B child tokens, leading to B^2 expansions as we have B branches, each with B children. From the B^2 expansions, we select the top B branches based on their joint probabilities for the next tree layer expansion. Upon reaching the maximum depth, we retain up to m tokens from the total $B + (D-1)B^2$ nodes, selecting those with the highest joint probability for verification.

2.3 Architecture of $\mathcal{M}_{\text{Small}}$

Speculative decoding architectures broadly fall into two categories, *independent* and *self-drafting*. Independent drafters are typically smaller versions of $\mathcal{M}_{\text{Large}}$ from the same model family (Li et al., 2024a; Zhao et al., 2024; He et al., 2023) while self-drafting methods leverage either a subset of $\mathcal{M}_{\text{Large}}$ and/or newly initialised parameters (Ankner et al., 2024; Cai et al., 2024a).

Our contribution is built on EAGLE (Li et al., 2024b), a *self-drafting* architecture which has shown the best results on the Spec-Bench (Xia et al., 2024) leaderboard so far. The drafter reuses the *token_embed* and *LM_head* parameters of $\mathcal{M}_{\text{Large}}$ (1). It takes as input the ground-truth activations of the last decoder layer of $\mathcal{M}_{\text{Large}}$, $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t$ and the tokens of the sequence $y_1, \cdots, y_t$ to predict the next activations $\hat{\boldsymbol{o}}_{t+1}$, which is passed to the *LM_head* to predict the next token distribution:

$$\hat{\boldsymbol{o}}_{t+1} = \mathcal{M}_{\text{Small}}^{\text{EAGLE}}((\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t),\ \textit{token\_embed}(y_1, \cdots, y_t)),\ \hat{y}_{t+1} \sim \text{Softmax}(\textit{LM\_head}(\hat{\boldsymbol{o}}_{t+1}))$$

The process is repeated by appending $\hat{\boldsymbol{o}}_{t+1}, \hat{y}_{t+1}$ to the inputs to auto-regressively draft tokens $\hat{y}_{t+2}$.

3 Methodology

3.1 Mixture of Attentions

We begin by defining important properties of $\mathcal{M}_{\text{Small}}$ followed by detailed architectural choices.

3.1.1 Partial Observability

In Markov Decision Processes (Kaelbling et al., 1998), partial observability is a common challenge where the agent does not have enough information about the true underlying state to take optimal decisions. This limitation can significantly degrade the agent's performance. Several approaches have been proposed to mitigate this, e.g., adding additional previous observations (Mnih et al., 2015). In drafting, it is important not to suffer from partial observability to draft future tokens more accurately. We extend this notion in our context with the following:

Property 3.1 (Partial observability)**.** *Given a ground truth function $F : \mathcal{X} \to \mathcal{Z}$, a drafter function $f : \mathcal{Y} \to \mathcal{Z}$ and an observation function $g : \mathcal{X} \to \mathcal{Y}$, such that for any $x \in \mathcal{X}$, f(g(x)) models F(x), f suffers from partial observability if g is non-injective: $\exists(x, x') \in \mathcal{X}^2, x \neq x', g(x) = g(x')$.*

We can observe that the EAGLE drafter suffers from partial observability when F is f_{Large}, f is $\mathcal{M}^{\text{EAGLE}}_{\text{Small}}$ and $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t = g(\boldsymbol{h}_{\leq t}, (y_1, \cdots, y_t))$. In other words, $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t$ is only a partial observation of the true state $(\boldsymbol{h}_{\leq t}, y_t)$ of the dynamic system (1), hindering the capacity of $\mathcal{M}_{\text{Small}}$ to predict the right tokens of $\mathcal{M}_{\text{Large}}$.

Layer Self-Attention Aiming to alleviate this, our new architecture takes as input the state of the dynamic system (1). However, $\boldsymbol{h}_{<t}$ is a large tensor of shape $(T, L, 2E_{kv})$ where T is the sequence length, L the number of layers in $\mathcal{M}_{\text{Large}}$, and E_{kv} the embedding size of the key and values. Therefore, we introduce Layer Self-Attention (LSA) followed by a mean aggregation operation to reduce its dimension to $(T, 2E_{kv})$ and extract the most relevant token information from every layer (Fig 1). Self-attention is performed over the layer dimension and each token is treated independently in this layer. During drafting, we have access to the past key values of all the layers, therefore, the attention mask of LSA is bidirectional/full, see Figure 2. We only perform the LSA computations *once at the start of each drafting phase*, see Appendix A.3 for a *detailed algorithm* of the information flow.

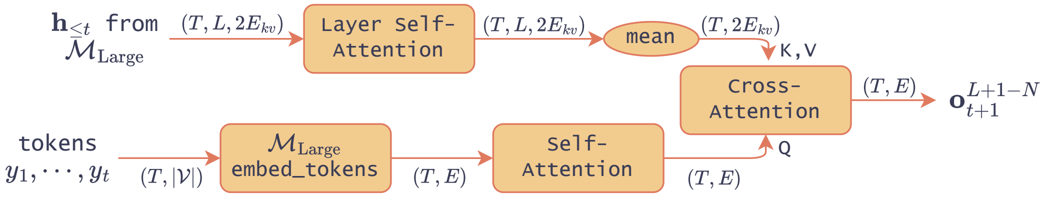


Figure 1: A schematic overview of the mixture of attentions information flow. Layer Self-Attention and mean aggregation are called *only once per drafting cycle*, i.e after each verification. New tokens are drafted auto-regressively using Self-Attention, updating only the Cross-Attention layer query.

3.1.2 Lack of on-policyness

Discrepancies between training and testing scenarios arise because, during training, transformer models are typically conditioned on ground-truth sequences, assuming that all previous inputs are correct. If this assumption seems unproblematic for the standard training of transformers, assuming it for training $\mathcal{M}_{\text{Small}}$ in speculative-decoding scenarios is much more delicate. It is known that some of the previous inputs are generated directly from $\mathcal{M}_{\text{Small}}$, therefore much less accurate. The more tokens we predict with $\mathcal{M}_{\text{Small}}$ only, the more error accumulation we can expect. To alleviate this, EAGLE adds uniform noise into its observations $(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t)$ at training time, but this is not ideal.

In order to train $\mathcal{M}_{\text{Small}}$ optimally, we need to ensure that its training and inference conditions are closely matched. Specifically, this means training $\mathcal{M}_{\text{Small}}$ as if some of the previous tokens were generated by itself. Additionally, we should account for situations where the activations from $\mathcal{M}_{\text{Large}}$ are not available, i.e. during the drafting cycle. This approach is called *on-policy* training. In on-policy training, the data used for training is generated by the same policy (or model) that is currently being trained. For example, when we train a transformer using next-token prediction on a static dataset, this is considered *off-policy* because the data doesn't change based on the model's decisions. However, if we mix this static dataset with data generated by the model itself during training, we move towards a more *on-policy* approach. Similarly, if the model won't have access to certain information, e.g. the activations of $\mathcal{M}_{\text{Large}}$ during generation, then always training $\mathcal{M}_{\text{Small}}$ with that information would also be considered *off-policy*.

However, on-policy training is very costly because we would need to generate from the model during training. To formalise this limitation, we introduce the concept of T-step boundedness:

Property 3.2 (T-step bounded)**.** *A drafter f is said to be T-*step bounded *if, in a single forward pass, it can predict up to T future tokens without additional input from $\mathcal{M}_{Large}$, i.e., $f(y_1, y_2, \ldots, y_t) \rightarrow (\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+T})$.*

This property is important to efficiently train the drafter. For instance, the EAGLE drafter is 1-step bounded. If one wanted to perform prediction at time $t + 2$, two forward passes would be required due to the auto-regressive layer that requires the previous $\hat{\boldsymbol{o}}_{t+1}$ as input, which would be very costly to train on-policy. By contrast, a drafter that is T-step bounded with $T > 1$ can predict multiple future tokens within a single forward pass.

Cross-Attention In order to make our drafter partly T-step bounded with $T > 1$, the main component of our architecture is a Cross-Attention (CA) layer where the query comes from the tokens and the key and values come from $\mathcal{M}_{\text{Large}}$ activations. More precisely, the key and values come from the output of LSA. Having input queries for time $t + 1$ to $t + K$ coming into the CA layer and keys-values from $\mathcal{M}_{\text{Large}}$ only up to time t effectively means the CA layer is K-step bounded. This allows us to train the CA layer more *on-policy* efficiently because it simulates what would happen during generation: we only have access to the activations from $\mathcal{M}_{\text{Large}}$ up to time t but still have to make prediction for up to time $t + K$. Note that it is still not fully *on-policy* yet as the input queries for time $t + 1$ to $t + K$ are not assumed to be generated from $\mathcal{M}_{\text{Small}}$. During training, multiple K are sampled to simulate different lengths of accepted drafts by changing the CA layer mask. For instance, in Figure 2, we have $K = 4$ followed by $K = 3$. On the contrary, during generation, we do not apply masking as we want to let $\mathcal{M}_{\text{Small}}$ attend all the currently available activations of $\mathcal{M}_{\text{Large}}$.

Self-Attention In order to motivate the introduction of a self-attention (SA) layer, we start by observing that the cross-attention layer is input-independent (3.3) w.r.t. the input queries, i.e one input query does not influence the results of another query.

Property 3.3 (Input-independence)**.** *A layer f is input-independent if for any choice of n inputs $\boldsymbol{x} = (x_1, \cdots, x_n)$, we have $f(\boldsymbol{x}) = (f(x_0), \cdots, f(x_n))$.*

Therefore, if the queries of the CA layer came directly from the embedded tokens $y_1, \cdots, y_t$, $\mathcal{M}_{\text{Small}}$ would not have been aware of previously drafted tokens. It would only know the previous token treated by $\mathcal{M}_{\text{Large}}$ and the most recent y_t. But, in order to make accurate predictions, $\mathcal{M}_{\text{Small}}$ needs to be aware of the previously drafted tokens. Hence, we introduce a causal self-attention layer on the queries to mitigate this problem, shown in Figure 1 and summarised in Table 1.

Table 1: Comparison of the properties of our new architecture.

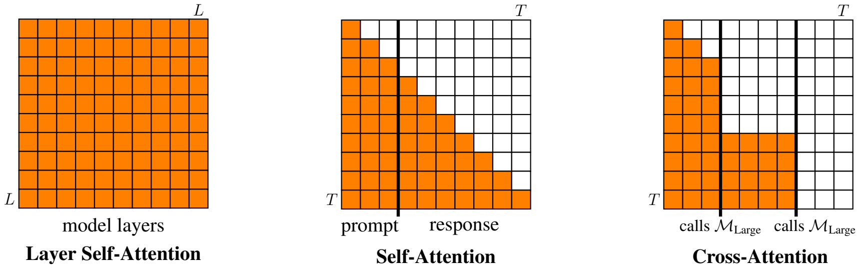
$\mathcal{M}_{\text{Small}}$	Autoregressive	T-step bounded	More on-policy	Observability
Ours	SA layer	variable T for CA layer	CA & LSA layers	LSA-enhanced
EAGLE-2	✓	1	✗	partial
Medusa	✗	fixed T	✗	partial

3.2 Target Layer Inference

Previous work assumed that the final hidden layer before *LM_head* was the most appropriate target (activations) $\mathcal{M}_{\text{Small}}$ should predict. However, we challenge that assumption by hypothesising that targeting a deeper $\mathcal{M}_{\text{Large}}$ layer may be more advantageous in terms of draft quality. We thus decompose the dynamic system (1) layer-by-layer by introducing l as the (superscript) layer index:

$$\begin{aligned} \boldsymbol{o}^1_{t+1} &= \textit{token\_embed}(y_t), & \boldsymbol{h}^l_{\leq t+1}, \boldsymbol{o}^{l+1}_{t+1} &= f^l_{\text{decoder}}(\boldsymbol{h}^l_{\leq t}, \boldsymbol{o}^l_{t+1}), \\ y_{t+1} &\sim \text{Softmax}(\textit{LM\_head}(\boldsymbol{o}^{L+1}_{t+1})) & l &= 1, \ldots, L \end{aligned}$$

where f^l_{decoder} is the decoder layer of $\mathcal{M}_{\text{Large}}$ at layer l. The state of this new dynamic system is composed of $(\boldsymbol{o}^l_{t+1}, \boldsymbol{h}^{<l}_{\leq t+1}, \boldsymbol{h}^{l\geq}_{\leq t})$. We observe that to perfectly predict $\boldsymbol{o}^{L+1}_{t+1}$, it is sufficient to perfectly predict $\boldsymbol{o}^L_{t+1}$ and *reuse* the f^L_{decoder} of $\mathcal{M}_{\text{Large}}$ and the already computed KV cache $\boldsymbol{h}^L_{\leq t}$ of the layer L at time t. The same recursive reasoning can be made to predict $\boldsymbol{o}^L_{t+1}$ from $\boldsymbol{o}^{L-1}_{t+1}$, etc.


L
L
model layers
T
T
prompt
response
T
T
calls $\mathcal{M}_{\text{Large}}$
calls $\mathcal{M}_{\text{Large}}$

Layer Self-Attention **Self-Attention** **Cross-Attention**

Figure 2: **Layer Self-Attention**: $\mathcal{M}_{\text{Large}}$ activations are transposed so that attention is computed over the layer dimension in order to aggregate token activations across layers. **Self-Attention:** The first 3 tokens represent the prompt, speculative decoding starts at token 4. **Cross-Attention:** Tokens 4 to 7 only attend to the prompt while tokens 8 to 10 attend to the first 7 tokens once $\mathcal{M}_{\text{Large}}$ was called for the second time allowing $\mathcal{M}_{\text{Small}}$ to use the activations from the newly verified tokens.

We assume (and later show) that predicting $\boldsymbol{o}^l_{t+1}$ is always easier than predicting $\boldsymbol{o}^k_{t+1}$ for $l < k$ due to $\boldsymbol{o}^l_{t+1}$ undergoing fewer layer transformations. Hence, we introduce a new hyperparameter TLI to refer to the target layer $\boldsymbol{o}^{L+1-\text{TLI}}$ that the $\mathcal{M}_{\text{Small}}$ should predict. When TLI > 0, the TLI last layers of $\mathcal{M}_{\text{Large}}$ (kept frozen during training) and their KV cache are used to output $\boldsymbol{o}^{L+1}$. Henceforth, we use notation (TLI $= l$) where l is an integer, to denote the target layer for inference. We can now provide the equation describing our $\mathcal{M}^{\text{Ours}}_{\text{Small}}$ for a given TLI assuming t was the last time we verified with $\mathcal{M}_{\text{Large}}$:

$$\begin{aligned}
\hat{\boldsymbol{o}}^{L+1-\text{TLI}}_{T+1} &= \mathcal{M}^{\text{Ours}}_{\text{Small}}(\boldsymbol{h}_{\le t}, \textit{token\_embed}(y_1, \cdots, y_t, \hat{y}_{t+1}, \cdots, \hat{y}_T)), \\
\hat{\boldsymbol{h}}^l_{T+1}, \hat{\boldsymbol{o}}^{l+1}_{T+1} &= f^l_{\text{decoder}}((\boldsymbol{h}^l_{\le t}, \hat{\boldsymbol{h}}^l_{>t,\le T}), \hat{\boldsymbol{o}}^l_{T+1}), \qquad l = L - \text{TLI}, \dots, L, \\
\hat{y}_{T+1} &\sim \text{Softmax}(\textit{LM\_head}(\hat{\boldsymbol{o}}^{L+1}_{T+1})).
\end{aligned}$$

3.3 Loss

Let $\mathcal{M}_{\text{Small}}$ be parameterised by $\boldsymbol{\theta}$, we use a similar training loss as EAGLE, i.e. a forward-KL loss, with a Smooth-L1 loss $\mathcal{L}$ between the predicted activations of the $\mathcal{M}_{\text{Small}}$ $\hat{\boldsymbol{o}}^{L+1-\text{TLI}}$ and the target one obtained from $\mathcal{M}_{\text{Large}}$:

$$\arg\min_{\boldsymbol{\theta}} \ \lambda_0 \text{KL}[\mathcal{M}_{\text{Large}} || \mathcal{M}_{\text{Small}}(\boldsymbol{\theta})] + \lambda_1 \mathcal{L}\left(\hat{\boldsymbol{o}}^{L+1-\text{TLI}}, \boldsymbol{o}^{L+1-\text{TLI}}\right). \tag{2}$$

To keep the training lightweight, we do not generate from $\mathcal{M}_{\text{Large}}$ or $\mathcal{M}_{\text{Small}}$ during training. This loss is only defined over the response part of the prompt of a fixed training dataset.

4 Experiments

In all experiments, we use *LLama3-8B-Instruct* (Dubey et al., 2024) as $\mathcal{M}_{\text{Large}}$. We train all $\mathcal{M}_{\text{Small}}$ on the Ultrachat dataset (Ding et al., 2023) without a system prompt and we do not assume that we know the system prompt at test time as it was observed that the training dataset can have a significant impact on the final performance (Yi et al., 2024). $\mathcal{M}_{\text{Small}}$ is trained with the standard Llama3-Instruct chat template. Ultrachat is composed of around 200k prompts with around 240M tokens using the LLama3 tokenizer. We use multiple test datasets for generation including various tasks such as reasoning, code generation, multi-turn conversation and summarisation. We notably relied on the SpecBench benchmark (Xia et al., 2024) and the following datasets: MT-Bench (Zheng et al., 2023), HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), Alpaca (Taori et al., 2023), CNN/Daily Mail (Nallapati et al., 2016) and Natural Questions (Kwiatkowski et al., 2019). We describe additional hyperparameters and experimental settings in Appendix A.1.

We compare our method to EAGLE-2 and an independent distilled $\mathcal{M}_{\text{Small}}$ of similar size (denoted ”Independent”). In order to train the EAGLE model, we assume TLI $= 0$ in the distillation loss (2). The independent $\mathcal{M}_{\text{Small}}$ leverages the *token_embed* and *LM_head* parameters of $\mathcal{M}_{\text{Large}}$ with only the decoder layers trained using an identical distillation loss (2) and $\lambda_1 = 0$. We do not compare to Medusa as EAGLE has consistently demonstrated superior speedups on various benchmarks (Xia et al., 2024). We also compare the performance of the official EAGLE-2 weights shared by Li et al. (2024b). We refer to this as "EAGLE-2 off.". Note that this model was trained on different data and with a fixed system prompt. We take care to match the number of model parameters, i.e. "Ours (N=0)", "EAGLE-2", "EAGLE-2 off." and "Independent 1.3B" all have 1.3B parameters (250M trainable and 1.05B frozen, for the "LM head" and "token embed" layers). We chose 250M trainable parameters to be directly comparable to EAGLE-2 and their official checkpoint. For tree decoding, we use a max breadth of 8, a depth of 6 and 62 max tokens to verify. We use float16 except for the attention softmax weights that are upscaled to float32.

We use standard metrics: *token-per-second* and *speedup ratios* to measure walltime improvements as well as hardware-independent metrics: average *acceptance length* τ (the average number of $\mathcal{M}_{\text{Small}}$ tokens accepted by $\mathcal{M}_{\text{Large}}$) and the *number of calls* to $\mathcal{M}_{\text{Large}}$.

4.1 Single Device

We now present the main single-device experiments using the SpecBench Xia et al. (2024) benchmark without a system-prompt to ensure a fair comparison between models.

Table 2: Speedup ratio and acceptance length τ on SpecBench using prompts from MT-Bench, HumanEval, GSM8K, Alpaca, Sum and QA datasets. Each model is fine-tuned for 30 epochs and uses EAGLE-2 tree decoding.

	Total	Trainable	MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Natural Ques.		Mean	
$\mathcal{M}_{\text{Small}}$	size	size	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Ours (TLI=3)	1.8B	250M	1.74	**4.65**	2.02	**5.41**	1.74	**4.65**	1.81	**4.80**	1.89	**5.04**	1.59	**4.23**	1.79	**4.79**
Ours (TLI=1)	1.55B	250M	**1.83**	4.19	**2.29**	5.30	**1.83**	4.19	2.02	4.65	2.04	4.74	1.71	3.94	**1.95**	4.50
Ours (TLI=0)	1.3B	250M	1.80	3.86	2.28	4.98	1.80	3.86	**2.03**	4.36	**2.10**	4.55	**1.72**	3.73	**1.95**	4.22
EAGLE-2	1.3B	250M	1.77	3.55	1.95	3.92	1.69	3.36	1.89	3.77	1.84	3.69	1.66	3.32	1.78	3.60
EAGLE-2 off.	1.3B	250M	1.75	3.52	2.06	4.15	1.80	3.60	1.70	3.37	1.60	3.19	1.38	2.75	1.71	3.43
Independent	1.7B	650M	1.50	3.63	1.91	4.64	1.26	3.01	1.57	3.81	1.56	3.78	1.72	3.94	1.58	3.80
Independent	1.3B	250M	1.23	3.50	1.50	4.36	0.95	2.70	1.33	3.79	1.28	3.59	1.10	3.13	1.23	3.51

Looking at Table 2, we can see that our Mixture of Attentions for SD achieves SOTA speedups when TLI $= 1$ and TLI $= 0$. Compared to EAGLE-2, we are on average 9.5% faster in terms of *tokens-per-second* generated. We also increase the acceptance length by 25% when $N = 1$. More single device experiments e.g. on the full HumanEval dataset are shown in Appendix A.4.

4.2 Client-Server

In this study, we investigate how *self-drafting* with our method performs in a client-server scenario. To do so, we place $\mathcal{M}_{\text{Small}}$ on a client device and host $\mathcal{M}_{\text{Large}}$ on a server (see Appendix A.2 for an illustration). The server is performing verification and sends the relevant $\mathcal{M}_{\text{Large}}$ activations to the client, which in turn is proposing new tokens. The server has 3 times more float16 tflops than the client. The devices are located in two different cities, separated by $\sim$300 km. The ping between the devices is around 9 ms and the bandwidth $\sim$50 Mbits/sec. In order to simulate a realistic client-server scenario, we are using 5G and 4G network profiles. In 4G, we assume a maximum of 20 Mbits/sec with a normally distributed delay of 21 ms ± 19 ms and a 0.1% chance of dropping packets. In 5G, we assume a normally distributed delay of 10 ms ± 10 ms with a 0.1% chance of dropping packets. To do so, we rely on the Linux traffic control subsystem.

In this scenario, the token-per-second performance also depends on the size of the messages. To this end, we analyse the length of the messages sent between the client and the server (see Table 7). There is a clear distinction between *self-drafting* methods that need to send/receive activation tensors and *independent* methods that only exchange text (e.g. token ids). Therefore, we shall analyse whether the improvement in drafting quality can offset the increase in message lengths. On the client, we encode each node in the draft tree using 3 bytes for the token id and 1 byte for its position in the tree. The server answers with the accepted tokens encoded using 3 bytes each plus the associated activations, if required. For Llama3-8B-Instruct and $N \leq 1$, our architecture's payload is less than or equal to EAGLE message lengths. In order to further reduce message sizes, we quantise the E and E_{kv} tensors to 8 bits. For both EAGLE and Mixture of Attentions, the initial message sent by the server (before the first token is drafted) is typically the biggest as it represents the activations of the entire prompt. Therefore, we additionally gzip-compress this message after quantisation.

Table 3: Performance on *HumanEval* with EAGLE-2 tree decoding under 5G and 4G profiles.

$\mathcal{M}_{\text{Small}}$	Total size	Trainable size	Tokens per second ↑		Acceptance length ↑	Calls $\mathcal{M}_{\text{Large}}$ ↓
			5G	4G		
Ours (TLI=3)	1.8B	250M	25.0	14.6	**4.99**	**20.8**
Ours (TLI=1)	1.55B	250M	30.6	20.3	4.68	22.5
Ours (TLI=0)	1.3B	250M	**34.1**	**25.1**	4.30	24.1
EAGLE-2	1.3B	250M	24.3	13.6	2.81	36.4
EAGLE-2 off.	1.3B	250M	28.6	15.0	3.51	29.5
Independent	1.7B	650M	28.5	23.7	3.73	27.1
Independent	1.3B	250M	18.3	16.1	3.16	32.4

In Table 3, we can observe that ”Ours (TLI=0)” achieves the fastest decoding speeds. Interestingly, it is even faster than independent small models that do not exchange any activation tensors. As expected, our Mixture of Attentions is not as fast as in the single device setting, but it can recover the speed of vanilla decoding in a single device setup (33 tokens-per-second, see Appendix A.4).

However, for this setting to be viable, just recovering the speed of vanilla decoding is not sufficient as it does not provide an advantage over an API call to $\mathcal{M}_{\text{Large}}$. Therefore, we show that our model can continue to generate the response by simulating a complete disconnection from the server.

Table 4: The success rate (pass@1, greedy decoding) on *HumanEval* in the event of an interrupted connection between the client and the server. EAGLE-2 tree decoding is used.

			A disconnection occurs after B new tokens.				
$\mathcal{M}_{\text{Small}}$	Total size	Trainable size	B = 1	B = 10	B = 25	B = 50	B = ∞
Ours (TLI=3)	1.8B	250M	2.48 %	**11.18 %**	18.01 %	**31.67 %**	45.9 %
Ours (TLI=1)	1.55B	250M	**3.10 %**	10.55 %	**21.11 %**	30.43 %	45.9 %
Ours (TLI=0)	1.3B	250M	2.48 %	9.31 %	19.2 %	29.81 %	45.9 %
EAGLE-2	1.3B	250M	0 %	8.07 %	16.77 %	27.32 %	45.9 %
EAGLE-2 off.	1.3B	250M	1.24 %	6.83 %	18.01 %	28.57 %	45.9 %
Independent	1.7B	650M	0 %	6.83 %	18.63 %	29.81 %	45.9 %
Independent	1.3B	250M	0 %	6.21 %	18.01 %	27.95 %	45.9 %
			Generation stops after B new tokens.				
Without local model (lower bound)			0 %	5.59 %	16.77 %	27.32 %	45.9 %

In Table 4, we can see that indeed, if a disconnection occurs, unlike API calls to $\mathcal{M}_{\text{Large}}$, we can continue to generate the response *right on the device*, i.e. complete additional correct solutions to competitive programming problems in HumanEval. Therefore, with an acceptable speed and the possibility to generate useful responses after a disconnection, we prove the viability of our proposed client-server setting, paving the way for a new framework for serving LLMs with small devices.

4.3 Ablation Study

We now present important ablation results for different components of our Mixture of Attentions architecture. Since multiple models were required to be fine-tuned for this study, we have limited each run to 10 epochs. For this ablation, we introduce the ”Ours (TLI=l, -LSA)” variant that does not rely on LSA and takes as input $\boldsymbol{o}_1, \cdots, \boldsymbol{o}_t$ as the keys and values of the CA layer. We also include two more EAGLE baselines, one with additional trainable parameters ”EAGLE (more params)” and another with additional decoder layers ”EAGLE (more layers)” but an equal number of trainable parameters. This is to ensure that the benefit of our architecture does not come from simply adding decoder layers or parameters. In this experiment, we use the HumanEval dataset with strict stopping criteria, exiting decoding as soon as the model no longer generates source code.

Table 5: An ablation study of our proposed architecture, tested on *HumanEval*. Each model is trained on $\sim$2.4B tokens. Chain (not tree) drafting with maximum 4 tokens is used for this study. The averages are computed over around 8500 drafting-verification cycles.

$\mathcal{M}_{\text{Small}}$	Total size	Trainable size	Tokens per second	Acceptance length (τ)
Ours (TLI=3)	1.8B	250M	39	**2.54**
Ours (TLI=1)	1.55B	250M	39	2.25
Ours (TLI=0)	1.3B	250M	**40**	2.14
Ours (TLI=1, -LSA)	1.55B	250M	21	1.28
Ours (TLI=1, -LSA, $\boldsymbol{o}_1, \cdots \boldsymbol{o}_t$ inputs)	1.55B	250M	36	2.04
Ours (TLI=0, -LSA, $\boldsymbol{o}_1, \cdots \boldsymbol{o}_t$ inputs)	1.3B	250M	38	1.93
EAGLE	1.3B	250M	30	1.45
EAGLE (more params)	1.45B	400M	29	1.28
EAGLE (more layers)	1.3B	250M	27	1.01

Does the on-policyness (brought with the CA layer) and the T-step bounded property have a positive impact on the quality of the drafts? In Table 5, we compare EAGLE with ”Ours (TLI=0, -LSA)” for an answer to this question. We can see that these components provide a major improvement of 26% in tokens-per-second as well as improved acceptance length of 33%.

How does partial observability influence the drafter *acceptance rate*? In Table 5, we can compare ”Ours (TLI=0, -LSA)” to ”Ours (TLI=0)” as well as ”Ours (TLI=1, -LSA)” to ”Ours (TLI=1)” and report that the tokens-per-second performance improves by 6% by introducing LSA, decreasing partial observability. Its impact is less crucial than the on-policyness brought by the CA layer.

Does increasing TLI increase the *acceptance rate*? Finally, by looking at the variation of TLI in Tables 2,3 and 5, increasing TLI also increases the acceptance length, as we hypothesised. However, this does not always have a positive impact on the tokens-per-second rate as it also increases the computational time of drafting. In the event of a complete disconnection in a client-server setting, however, a higher TLI will improve the quality of responses, which is something to consider when deploying Mixture of Attentions for SD on mobile devices.

5 Related Work

Medusa (Cai et al., 2024a) is one of the earliest works leveraging the activations of $\mathcal{M}_{\text{Large}}$ as inputs to $\mathcal{M}_{\text{Small}}$ for the purpose of SD. Thanks to their work, speculative decoding can be applied to any LLM by distilling an $\mathcal{M}_{\text{Small}}$. It generates K future tokens in parallel by training K new *LM_heads* where each head predicts a token at position $k \in K$ (Gloeckle et al., 2024). It was later extended by Kim et al. (2025) by refining the block drafts using task-independent n-gram and neural language models as lightweight rescorers. EAGLE (Li et al., 2024c) and Hydra (Ankner et al., 2024) are auto-regressive extensions of Medusa. They observe that non-auto-regressive generation limits the acceptance length as $\mathcal{M}_{\text{Small}}$ is not aware of previous tokens. We do not compare to Medusa or Hydra as EAGLE is ranked higher on the SpecBench leaderboard.

Tandem Transformers (Nair et al., 2024) propose an effective integration of $\mathcal{M}_{\text{Large}}$ and $\mathcal{M}_{\text{Small}}$ by letting $\mathcal{M}_{\text{Small}}$ attend to the down-projected hidden states of $\mathcal{M}_{\text{Large}}$. These rich contextualised representations enable $\mathcal{M}_{\text{Small}}$ to draft hypotheses with a higher acceptance rate as the two models are aligned on shared hidden states. We were not able to compare with them because of the lack of open-source implementation, the use of closed-source LLMs and an undisclosed amount of data/compute to reproduce the work. Moreover, tandem transformers appear to have a high communication overhead between big and small models, making it unrealistic for a client/server setting.

Orthogonal to our work, researchers have recently proposed *training-free* SD methods. Lookahead Decoding (Fu et al., 2024) generates new tokens with a single $\mathcal{M}_{\text{Large}}$ using Jacobi iterations, extended by CLLM Kou et al. (2024) and Ouroborous (Zhao et al., 2024). We evaluated the latter in our settings, however, it was shown to be less efficient than the EAGLE-2 tree decoding strategy, see Appendix A.4. For additional related and orthogonal work in the extended SD landscape, we refer the reader to Xia et al. (2024) for a detailed and highly informative speculative decoding survey.

Du et al. (2024) previously proposed to leverage the KV-cache of some layers of $\mathcal{M}_{\text{Large}}$. They do not justify why using the KV-cache instead of the output of each layer, nor how to exactly choose which layer to include as input of $\mathcal{M}_{\text{Small}}$. However, with our dynamical system point of view, we showed that the KV-cache of all the layers is part of the state. The introduction of LSA allows to exploit it in its whole with a limited number of layers, whereas Du et al. (2024) would need to have the same number of layers in $\mathcal{M}_{\text{Small}}$ and $\mathcal{M}_{\text{Large}}$ to fully capture it, resulting in a slow drafting speed.

Although we focused on improving the current SOTA method (EAGLE-2), our observations (partial observability, on-policyness and target inference layer) are true for many self-drafting methods, for instance, it could also be applied to Medusa (Cai et al., 2024a), MLP Speculator (Wertheimer et al., 2024) or Gloeckle et al. (2024); Kim et al. (2025).

Regarding non-self-drafting SD, it should be studied on a case-by-case basis. For instance, target inference layer could potentially be applied to independent small models. Many student-teacher distillation frameworks (Gu et al., 2024; Zhou et al., 2023), already leverage the on-policyness property by generating directly from the student but are mostly are 1-step bounded (therefore expensive to train). For SD methods based on lookahead decoding, it would generally not apply. One exception is Ouroboros (Zhao et al., 2024) that leverages a small model with lookahead decoding. Their small model could also benefit from our solutions.

6 Conclusion

We have introduced a Mixture of Attentions architecture for Speculative Decoding to effectively address several limitations of existing state-of-the-art methods. In order to enhance drafting accuracy of $\mathcal{M}_{\text{Small}}$, we proposed a mixture of attention layers: Layer Self-Attention to mitigate partial observability and Self-Attention followed by Cross-Attention to train more on-policy. We have then introduced Target Layer Inference, a novel method that lets $\mathcal{M}_{\text{Small}}$ reuse the last N layers of $\mathcal{M}_{\text{Large}}$, enabling a trade off between the drafting speed and accuracy. Experimental results show that we achieve state-of-the-art decoding speedups in the standard single-device setup, improving over EAGLE-2 by 9.5% and extending acceptance lengths by up to 25%. We have also introduced a client-server paradigm and demonstrated that our *self-drafting* speculative decoding method is a viable alternative to API calls to $\mathcal{M}_{\text{Large}}$. Under this paradigm, the client can continue to generate responses with the highest accuracy and speed after a complete disconnection from the network. As a future direction, it would be interesting to investigate whether N could be predicted by $\mathcal{M}_{\text{Small}}$ to automatically balance speed and accuracy depending on the current network conditions.

REFERENCES

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.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024a.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads, 2024b. URL https://arxiv.org/abs/2401.10774.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang

Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Miniplm: Knowledge distillation for pre-training language models. *arXiv preprint arXiv:2410.17215*, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *arXiv preprint arXiv:2405.19715*, 2024.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Taehyeon Kim, Ananda Theertha Suresh, Kishore Papineni, Michael D Riley, Sanjiv Kumar, and Adrian Benton. Accelerating blockwise parallel language models with draft refinement. *Advances in Neural Information Processing Systems*, 37:34294–34321, 2025.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. Aligning large language models with representation editing: A control perspective, 2024. URL https://arxiv.org/abs/2406.05954.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models. *arXiv preprint arXiv:2403.00835*, 2024.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

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

Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. Nearest neighbor speculative decoding for llm generation and attribution. *arXiv preprint arXiv:2405.19325*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty, 2024c. URL `https://arxiv.org/abs/2401.15077`.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and Roy Schwartz. Accelerating speculative decoding using dynamic speculation length. *arXiv preprint arXiv:2405.04304*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Pranav Ajit Nair, Yashas Samaga, Toby Boyd, Sanjiv Kumar, Prateek Jain, Praneeth Netrapalli, et al. Tandem transformers for inference efficient llms. *arXiv preprint arXiv:2402.08644*, 2024.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Davis Wertheimer, Joshua Rosenkranz, Thomas Parnell, Sahil Suneja, Pavithra Ranganathan, Raghu Ganti, and Mudhakar Srivatsa. Accelerating production llms with combined token/embedding speculators. *arXiv preprint arXiv:2404.19124*, 2024.

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*, pp. 3909–3925, 2023.

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.

Euiin Yi, Taehyeon Kim, Hongseok Jeung, Du-Seong Chang, and Se-Young Yun. Towards fast multilingual llm inference: Speculative decoding and specialized drafters. *arXiv preprint arXiv:2406.16758*, 2024.

Biao Zhang, Ivan Titov, and Rico Sennrich. Sparse attention with linear units. *arXiv preprint arXiv:2104.07012*, 2021.

Weilin Zhao, Yuxiang Huang, Xu Han, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. Ouroboros: Speculative decoding with large model enhanced drafting. *arXiv preprint arXiv:2402.13720*, 2024.

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*, 36:46595–46623, 2023.

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.

A Appendix

The source code is publicly available at https://github.com/huawei-noah/HEBO/tree/mixture-of-attentions/.

A.1 Hyperparameters

Table 6: List of our hyperparameters.

Distillation	
Learning rate for gradient descent	$3 \cdot 10^{-5}$
Total numbers of transformer updates	186000
Minibatch size	32
Mixed-precision training	yes, float16
Weight of reserve KL loss (λ_0)	0.1
Weight of L1 smooth loss (λ_1)	1.0
L2 gradient clipping	1.0
T-step bounded mask for the CA layer	Uniform between 5 to 15
Architecture	
Number of layers L of $\mathcal{M}_{\text{Large}}$	32
Embedding dimension E of $\mathcal{M}_{\text{Large}}$	4096
Embedding dimension of keys and values E_{kv} of $\mathcal{M}_{\text{Large}}$	1024
Dropout rate	0.0
Embedding dimension of Layer Self-Attention	2048
Embedding dimension of Self-Attention	4096
Embedding dimension of Cross-Attention	4096
Size of the MLP projection after Layer Self-Attention	6144
Size of the MLP projection after Self-Attention	512
Size of the MLP projection after Cross-Attention	7168
Embedding dimension of keys and values of Layer Self-Attention	1024
Embedding dimension of keys and values of Self-Attention	512
Embedding dimension of keys and values of Cross-Attention	1024

A.2 Client Server Deployment

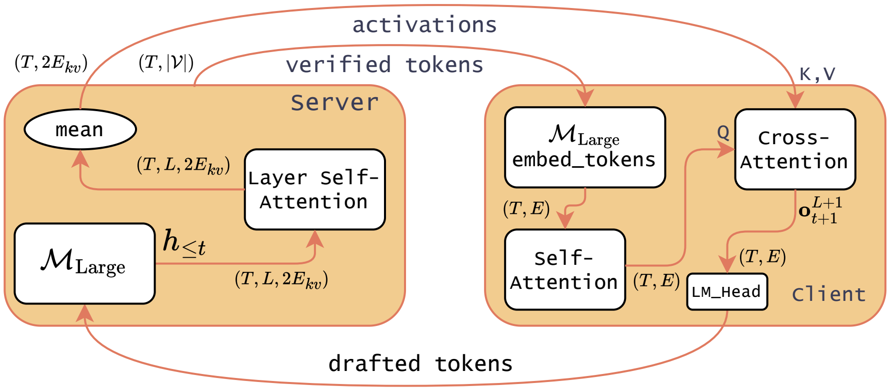


Figure 3: A client-server setting for our mixture of attentions architecture with $N = 0$.

Table 7: The size of the message (before quantisation) in bytes. M = number of nodes in the draft tree, A = number of accepted tokens, E = hidden size, E_{kv} = hidden size of key and query vectors.

$\mathcal{M}_{\text{Small}}$	Sent by Client	Sent by Server
Ours	$4M$	$3A + 2AE_{kv}(\text{TLI} + 1)$
EAGLE	$4M$	$3A + AE$
Independent	$4M$	$3A$

A.3 Algorithm

Algorithm 1 Generation algorithm for $\mathcal{M}^{\text{Ours}}_{\text{Small}}$ assuming chain decoding

```
Require: Input sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_t)$, draft length $K$, target layer inference TLI
Obtain $\boldsymbol{h}_{\le t}$ activations and $y_{t+1}$ with a forward pass in $\mathcal{M}_{\text{Large}}$ given input $\boldsymbol{y}$
$\boldsymbol{y} \leftarrow (\boldsymbol{y}, y_{t+1})$
$\boldsymbol{kv} \leftarrow$ LSA_layer_with_mean $(\boldsymbol{h}_{\le t})$
while stopping criteria is not meet on $\boldsymbol{y}$ do
    for $i = 1$ to $K$ do
        $\boldsymbol{q} \leftarrow$ SA_layer(token_embed($\boldsymbol{y}$))
        $\hat{\boldsymbol{o}}^{L+1-N} \leftarrow$ CA_layer($\boldsymbol{q}, \boldsymbol{kv}$)
        if $N > 0$ then
            for $l = L - N$ to $L$ do
                $[\hat{\boldsymbol{h}}^l, \hat{\boldsymbol{o}}^{l+1}] \leftarrow f^l_{\text{decoder}}((\boldsymbol{h}^l_{\le t}, \hat{\boldsymbol{h}}^l_{>t,\le t+i}), \hat{\boldsymbol{o}}^l)$
            end for
        end if
        $\hat{y} \sim$ Softmax($LM\_head(\hat{\boldsymbol{o}}^{L+1})$)
        $\boldsymbol{y} \leftarrow (\boldsymbol{y}, \hat{y})$
    end for
    Identify $K'$ verified tokens out of the $K$ latest tokens of $\boldsymbol{y}$, obtain associated $\boldsymbol{h}'$ and obtain $y'$ with a forward pass in $\mathcal{M}_{\text{Large}}$ with inputs $\boldsymbol{y}_{|\boldsymbol{y}|-K,\cdots,|\boldsymbol{y}|}$ and $\boldsymbol{h}_{\le t}$
    $\boldsymbol{kv'} \leftarrow$ LSA_layer_with_mean $(\boldsymbol{h}')$
    $\boldsymbol{kv} \leftarrow (\boldsymbol{kv}, \boldsymbol{kv}')$
    Update $\boldsymbol{h}$ by appending the new $\boldsymbol{h}'$ components
    Discard previous $\hat{\boldsymbol{h}}$
    $\boldsymbol{y} \leftarrow \boldsymbol{y}_{1,\cdots,|\boldsymbol{y}|-K+K'}$ (keep only the verified tokens)
    $t \leftarrow |\boldsymbol{y}|$
    $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y')$
end while
return $\boldsymbol{y}$
```

A.4 Additional Experiments

Accuracy of the generated text We ran several experiments to assess the quality of the generated responses using greedy decoding. We focused on 3 datasets from SpecBench (HumanEval, GSM8K and CNN/DM) that do not require access to proprietary models/APIs for evaluation (llm-as-a-judge).

Table 8: Quality of the generated text.

Vanilla decoding	HumanEval (pass@1)	GSM8K (accuracy)	CNN/DM (Rouge-L f-score)
Llama3-8B-Instruct	62.5%	80%	0.3071
Speculative Decoding	**HumanEval (pass@1)**	**GSM8K (accuracy)**	**CNN/DM (Rouge-L f-score)**
Ours (TLI=3)	62.5%	80%	0.3053
Ours (TLI=1)	62.5%	81.25%	0.3068
Ours (TLI=0)	62.5%	80%	0.3070
EAGLE-2	62.5%	81.25%	0.3062
EAGLE-2 off	62.5%	80%	0.3056
Independent 1.7B	62.5%	80%	0.3067
Independent 1.3B	62.5%	80%	0.3064

We report the results in Table 8. The pass@1 on HumanEval is the same across all methods. The accuracy on GSM8K actually improves w.r.t the base model on one question for Ours (TLI=1) and EAGLE-2. Finally, the ROUGE scores are also extremely similar, leading us to conclude that any differences to the base model are negligible and almost certainly appear due to using float16.

Qwen2.5 3B We trained 3 additional small models on the Ultrachat dataset to accelerate Qwen2.5 3B. EAGLE recommends to use one decoder layer of the big LLM to define the size of the small LM, which leads to a trainable size of 80M parameters. We kept the shared ”embed_tokens/LM_head” layer frozen.

Table 9: Speedup ratio and acceptance length τ on SpecBench using prompts from MT-Bench, HumanEval, GSM8K, Alpaca, Sum and QA datasets with Qwen2.5-3B Instruct.

	Total	Trainable	MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Natural Ques.		Mean	
$\mathcal{M}_{\text{Small}}$	size	size	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Ours (TLI=0)	0.4B	80M	**1.71**	**3.72**	**2.18**	**4.76**	**1.60**	**3.46**	**1.88**	**4**	**1.78**	**3.89**	**1.68**	**3.59**	**1.80**	**3.9**
EAGLE-2	0.4B	80M	1.59	3.2	1.84	3.70	1.53	3.06	1.81	3.54	1.60	3.23	1.62	3.17	1.66	3.31
Independent	0.4B	80M	1.59	3.37	2.04	4.38	1.44	3.03	1.70	3.52	1.54	3.27	1.50	3.12	1.63	3.44

Higher batch size with vLLM We implemented our approach in vLLM (Kwon et al., 2023) without tree decoding to support higher batch sizes and continuous batching.

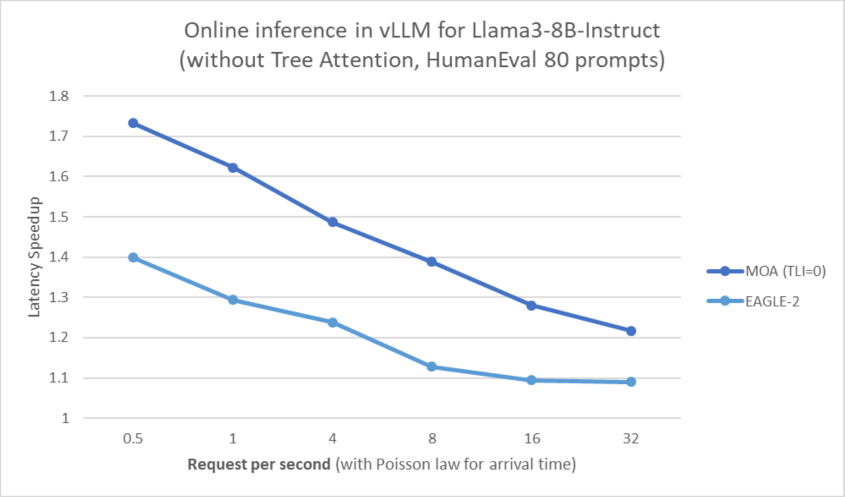


Figure 4: vLLM inference with continuous batching.

HumanEval in single device To perform this experiment, we reuse the same full HumanEval dataset with a strict stopping criteria as done in the ablation study in the single device setting.

Table 10: Test on *Human Eval*, each model is trained for 30 epochs.

$\mathcal{M}_{\text{Small}}$	Decoding	Total size	Trainable size	Tokens per second	Acceptance length (τ)
Ours (TLI=3)	EAGLE-2	1.8B	250M	54	**5.02**
Ours (TLI=1)	EAGLE-2	1.55B	250M	**58**	4.70
Ours (TLI=0)	EAGLE-2	1.3B	250M	57	4.30
EAGLE	EAGLE-2	1.3B	250M	43	2.82
EAGLE off.	EAGLE-2	1.3B	250M	52	3.50
Independent	EAGLE-2	1.7B	650M	46	3.72
Independent	EAGLE-2	1.3B	250M	34	3.17
Independent	Ouroboros	1.7B	650M	39	2.37
Baseline	Vanilla	-	-	33	1

From Table 10, we can observe we are 26% faster than EAGLE/EAGLE-2. We are also faster than independent small models and Ouroboros (Zhao et al., 2024).

A.5 Complexity Analysis

Let us analyze the standard decoder-only transformers doing vanilla decoding:

- in the first prefill stage, it grows in $\mathcal{O}(LKE(E+K))$ given we have L self-attention layers with K input tokens and an embedding size of E
- for the K’ new decoded tokens, it grows in $\mathcal{O}(\sum_i^{K'} L(E^2 + E(K+i))) = \mathcal{O}(LE(EK' + KK' + K'^2))$.

If we assume E and L are fixed, it grows in $\mathcal{O}((K+K')^2)$ overall. For speculative decoding, the first prefill stage is the same. Assuming S tokens are verified at a time, the verification would grow in $\mathcal{O}(\sum_i^{\frac{K'}{S}} L(SE^2 + SE(K+i))) = \mathcal{O}(LE(EK' + KK' + K'^2))$, leading to the same complexity as vanilla decoding. It dominates the complexity of self-drafting, but we can still analyse it. For EAGLE, decoding a new token grows in $\mathcal{O}(E^2 + EK)$ as it is a single self-attention layer. For our Mixture of Attentions architecture, the Self-Attention and Cross-Attention layers also grow in $\mathcal{O}(E^2 + EK)$. The Layer-Self Attention is only called once after every verification stage, so not at every decoding step, it grows in $\mathcal{O}(ALE_{kv}^2 + AE_{kv}L^2)$ if A is the number of accepted tokens in the previous phase. In our experiments, if we look at the first term, ALE_{kv}^2 is smaller than $number\_of\_decoded\_tokens \times E^2$ as E_{kv} is 4 times smaller than E, L is 32, A is in average 4.5 and $number\_of\_decoded\_tokens$ is 48. Similarly for the second term, $AE_{kv}L^2$ is usually smaller than $number\_of\_decoded\_tokens \times EK$ as soon as the request contains more than 24 tokens. Therefore, the time complexity is the same as EAGLE overall.

A.6 Privacy application

Another advantage of the client-server setup is that we can selectively ensure privacy for the client by only sending the non-sensitive part of the prompt to the server. Essentially, the client can split their input into a consecutive ”safe” text and a ”private” text. The server processes only the ”safe” text, which could be general context or non-sensitive information. The client keeps the ”private” text, such as confidential data or sensitive instructions, and handles this part locally with $\mathcal{M}_{\text{Small}}$.

For instance, the client might send the server some Python code along with a general description. However, any sensitive information, such as the login and password to inject into the code, remains on the client side and is not transmitted to the server. It is only passed to $\mathcal{M}_{\text{Small}}$. This approach leverages the activations of $\mathcal{M}_{\text{Large}}$ to increase the accuracy of $\mathcal{M}_{\text{Small}}$ for parts of the task while ensuring that sensitive information is never exposed outside the client’s environment.