# Dynamic-Width Speculative Beam Decoding for LLM Inference

**Zongyue Qin, Zifan He, Neha Prakriya, Jason Cong, Yizhou Sun**

University of California Los Angeles, CA, USA
{qinzongyue, zifanhe, nehaprakriya, cong, yzsun}@cs.ucla.edu

## Abstract

Large language models (LLMs) based on transformer architecture have shown outstanding performance across numerous real-world tasks. However, the autoregressive nature of these models makes the inference process slow and costly. Speculative decoding has emerged as a promising solution, leveraging a smaller auxiliary model to draft future tokens, which are then validated simultaneously by the larger model, achieving a speed-up of 1-2×. Although speculative decoding matches the same distribution as multinomial sampling, multinomial sampling itself is prone to suboptimal outputs, whereas beam sampling is widely recognized for producing higher-quality results by maintaining multiple candidate sequences at each step. This paper explores the novel integration of speculative decoding with beam sampling. However, there are four key challenges: (1) how to generate multiple sequences from the larger model's distribution given draft sequences from the small model; (2) how to dynamically optimize the number of beams to balance efficiency and accuracy; (3) how to efficiently verify the multiple drafts in parallel; and (4) how to address the extra memory costs inherent in beam sampling. To address these challenges, we propose dynamic-width speculative beam decoding (DSBD). Specifically, we first introduce a novel draft and verification scheme that generates multiple sequences following the large model's distribution based on beam sampling trajectories from the small model. Then, we introduce an adaptive mechanism to dynamically tune the number of beams based on the context, optimizing efficiency and effectiveness. Besides, we extend tree-based parallel verification to handle multiple trees simultaneously, accelerating the verification process. Finally, we illustrate a simple modification to our algorithm to mitigate the memory overhead of beam sampling. Experimental results show that our approach achieves a 1.5-1.9× speed-up and 1.8-2.5× lower energy consumption compared to beam sampling, with no loss in downstream performance. Moreover, it can produce significantly higher-quality outputs than speculative decoding, while maintaining similar time, memory, and energy costs. In summary, our method offers a more efficient and effective inference process for LLMs.

## 1 Introduction

In recent years, large language models based on transformer architecture (Vaswani et al. 2017), such as GPT-4 (Achiam et al. 2023), Llama-3 (AI@Meta 2024), and PALM (Anil et al. 2023), have demonstrated remarkable performance across a wide range of real-world tasks, including text generation, summarization, and translation. However, the autoregressive nature of these models, where tokens are generated one at a time, leads to slow inference speeds and high computational costs. As the size and complexity of LLMs continue to increase, the demands on computational resources and energy consumption during inference have become major concerns, limiting their scalability and accessibility.

Speculative decoding has emerged as a promising technique to accelerate LLM inference by leveraging a smaller auxiliary model to generate draft tokens. These tokens are then validated by the large model, resulting in a significant reduction in inference time. The primary advantage of speculative decoding is its ability to maintain the same quality of output as multinomial sampling while achieving a 1-2× speed-up. However, multinomial sampling itself is limited to generating a single sequence based on local optimality. This limitation makes it susceptible to returning suboptimal results, as it lacks the diversity that could be achieved by considering multiple candidate sequences simultaneously.

Motivated by the need to improve the output quality, we explore the integration of speculative decoding with beam sampling, a technique that maintains multiple candidate sequences (beams) at each step to enhance the diversity and quality of the generated output. This fusion, however, presents several challenges. First, while previous studies focused on obtaining a single token from the large model distribution given draft tokens from the smaller model, our approach requires generating multiple tokens (beams) simultaneously, which necessitates a new verification scheme. Second, determining the optimal number of beams is critical: too many beams can lead to inefficiency due to a high rejection rate, while too few beams may result in under-utilization of the small model's potential and low effectiveness. Third, efficiently verifying multiple draft sequences in parallel requires a technique that can process and validate multiple beams concurrently. Fourth, addressing the additional memory cost of storing multiple key-value caches is crucial to enable LLMs to use beam sampling in practice.

To address these challenges, we propose dynamic-width speculative beam decoding (DSBD) that combines speculative decoding with beam sampling through a series of inno-

vations. First, we introduce a draft and verification scheme that processes beam decoding trajectories as forests of trees, which are verified layer by layer by the large model. This approach allows us to efficiently generate multiple beams while maintaining the large model's sampling distribution. Second, we propose a mechanism to dynamically adjust the number of beams based on the context, ensuring a balance between efficiency and effectiveness. Third, we extend existing tree-based parallel verification techniques (Miao et al. 2023) to operate on multiple trees, incorporating a forest-based parallel verification strategy that enhances the speed of the verification process. Finally, we introduce a simple modification to DSBD that reduces the memory cost by storing only one set of key-value caches, while still delivering better output quality than multinomial sampling.

Experimental results show that our approach achieves a 1.5-1.9× speed-up and 1.8-2.5× smaller energy consumption than beam sampling, without sacrificing performance on downstream tasks. Besides, it can produce significantly higher-quality outputs than speculative decoding, while maintaining comparable time, memory, and energy costs. These findings suggest that DSBD successfully bridges the gap between speculative decoding and beam sampling, providing a more efficient and effective decoding method for LLMs. Our code is open source[1].

# 2 Preliminaries

## 2.1 Decodings of LLMs

Let $p$ denote the distribution defined by a large language model $M_p$. Given an input prefix, the optimal decoding algorithm is to generate a sequence of $N$ tokens with maximum likelihood $p(x_{1:N}|input)$.

**Multinomial Sampling.** Multinomial sampling, also known as standardized sampling, samples the next token $x_t$ based on $\mathcal{T} \circ p(\cdot|x_{1:t-1}, input)$, where $\mathcal{T}$ is a warping operation applied to enhance the high probability region. Some common warping operations include *top-k* warping, which limits the selection to the top $k$ tokens, and *top-p* warping, where tokens are sampled from the smallest possible subset of the vocabulary whose cumulative probability mass exceeds a specified threshold $p$. The deterministic version of multinomial sampling is a special case when $k = 1$.

**Beam Sampling.** Beam decoding aims to do a better job than multinomial sampling. For each position $t$ ($1 \leq t \leq N$), it maintains $W > 1$ candidate sequences, which are also called *beams*. Assume we have already kept the $W$ sequences $\mathcal{I}_{t-1} = \{x^{(1)}_{1:t-1}, \ldots, x^{(W)}_{1:t-1}\}$ at position $t-1$, $W$ sequences with length $t$ are then sampled from $\mathcal{T} \circ p_{beam}$, where $p_{beam}$:$\mathcal{I}_{t-1} \times V \to [0, 1]$ is the beam sampling probability:

$$p_{beam}(x^{(i)}_{1:t-1}, x_t) = \frac{p(x^{(i)}_{1:t-1}, x_t|input)}{\sum_{x^{(j)}_{1:t-1}, x'_t \in \mathcal{I}_{t-1} \times V} p(x^{(j)}_{1:t-1}, x'_t|input)} \quad (1)$$

[1]https://github.com/ZongyueQin/DSBD

Notice that $p(x^{(i)}_{1:t-1}, x_t|input) = p(x_t|x^{(i)}_{1:t-1}, input) \cdot p(x^{(i)}_{1:t-1}|input)$. In practice, beam sampling stores the likelihood $p(x^{(i)}_{1:t-1}|input)$ for each beam, and the computation complexity of $p_{beam}$ is $O(W \cdot |V|)$. In deterministic beam sampling, the top $W$ sequences with the highest likelihood $p_{beam}(x_{1:t})$ will be kept.

(Shi et al. 2024) shows that beam sampling in general has better downstream effectiveness than multinomial sampling. Figure 1 shows an example where beam decoding returns a better output.

## 2.2 Vanilla Speculative Decoding

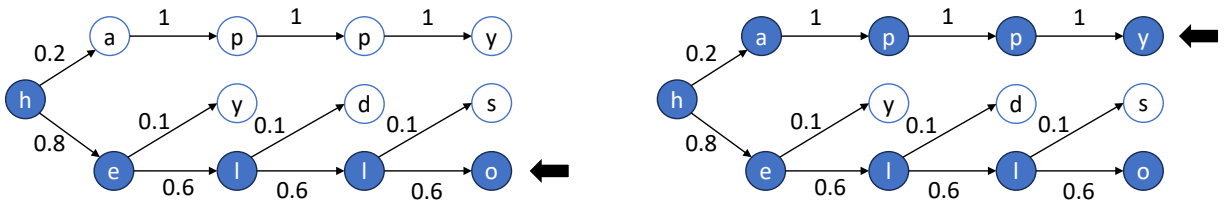
Speculative decoding utilizes a small model to generate the next $\gamma$ tokens and then employs the large model to verify these drafted tokens *in parallel*. The process is summarized as follows:

1. Given $input$, the small model samples $\gamma$ draft tokens $x_1, \ldots, x_\gamma$ using greedy decoding, based on the warped predicted conditional probability $\tilde{q}(x_t|x_{1:t-1}, input)$ for $t = 1, \ldots, \gamma$, where $\tilde{q} = \mathcal{T} \circ q$ and $q$ is the small model's output distribution.
2. The large model verifies the draft tokens in parallel by computing the conditional probability $\tilde{p} = \mathcal{T} \circ p(x_t|x_{1:t-1}, input)$ for $t = 1, \ldots, \gamma$.
3. Each draft token $x_t$ is accepted with probability $\min(1, \tilde{p}(x_t)/\tilde{q}(x_t))$. The draft tokens before the first rejected token are kept as the decoding output. An additional token is sampled from a residual distribution as a correction for the first rejected token. The accepted tokens and the resampled token are then appended to the context $prefix$ as input for the next iteration.
4. Repeat steps 1-3 until reaching the stopping criteria, such as a length limit.

By verifying $\gamma$ tokens in parallel with one run of the large model, speculative decoding reduces the time cost compared to calling the large model $\gamma$ times. Additionally, although the small model still runs in an autoregressive manner, its inference speed is much faster than the large model. This makes speculative decoding an effective method to accelerate the inference process of LLMs. Moreover, it has been proven that each token $x_t$ generated by speculative sampling follows the identical sampling distribution as multinomial sampling.

# 3 Methodology

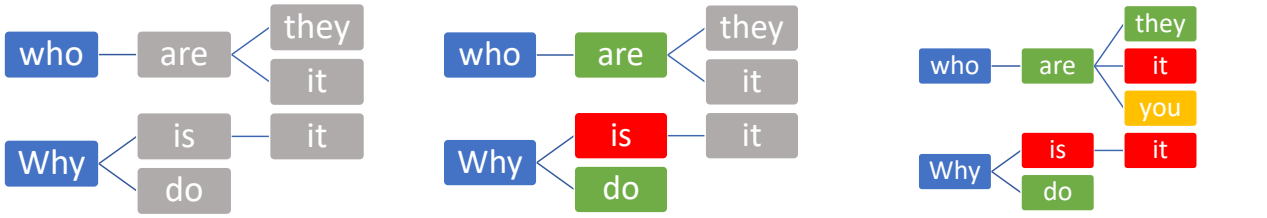
The primary goal of our method is to enhance the efficiency and effectiveness of large language model (LLM) inference by combining the speed advantages of speculative decoding with the accuracy and diversity benefits of beam sampling. We first introduce a novel draft and verification scheme that keeps identical distribution as beam sampling. Then, we describe an adaptive beam management strategy. Next, we illustrate a forest-based parallel verification mechanism. Finally, we discuss how to resolve the additional memory cost inherent in beam sampling.




(a) Multinomial sampling.

(b) Beam sampling with 2 beams.

Figure 1: Examples of greedy and beam sampling. Some nodes are omitted in the figures. Assume the sampling probability is warped to always sample the tokens with the largest probabilities. Given $prefix$ "`h`", multinomial sampling generates "`hello`" with an average perplexity of 1.55. Beam sampling generates "`happy`" with an average perplexity of 1.49.




(a) Draft forest from the small model.

(b) Verification result of the 1st draft layer.

(c) Verification result of the 2nd draft layer.

Figure 2: Illustration of one iteration of Speculative Beam Decoding. (a) Draft Stage: given the input beams "*who*" and "*why*", the small model first generates a trace of beam sampling. (b)(c): Verification Stage. When verify the first draft layer, "*who are*" and "*why do*" are accepted, while "*why is*" is rejected. When verify the second draft layer, "*why is it*" is directly rejected because its parent is rejected. Then "*who are they*" is accepted, while "*who are it*" is rejected. And another beam "*who are you*" is sampled from the residual distribution.

### 3.1 Draft and Verification Scheme

**Overview** As illustrated in Figure 2, the core idea of our method is to leverage a smaller, auxiliary model to generate multiple draft sequences, referred to as draft beams, which are then verified and refined by the larger model. This approach enables us to maintain multiple candidate sequences throughout the decoding process, thereby achieving better output quality than multinomial sampling, while improving the overall efficiency of beam sampling.

For now, assume that the number of beams (also referred to as the width, denoted as $W_L$) is fixed. In each iteration of our method, the input consists of the beams generated in the previous iteration. For the first iteration, the input is the initial input context. At each iteration, our method first uses the small model to perform beam sampling with a width of $W_S$ for $\gamma$ steps. Notice that we want $W_S > W_L$ because some draft beams might be rejected later. As illustrated in Figure 2a, it generates a trajectory that can be represented as a forest consisting of $W_L$ trees, which we refer to as the "draft forest". In this forest, each tree originates from an input beam, with the maximum depth of each tree being $\gamma + 1$. Starting from the second layer, each layer of the forest contains $W_S$ nodes, representing the intermediate beams at each step of the beam sampling process.

Once the draft forest is generated, our method leverages the large model to predict the distribution for the next token of each node (beam) in parallel. Using these distributions, DSBD then verifies each layer of the draft forest sequentially. For each layer, it calculates the joint probability of the beams and sequentially determines whether each beam should be accepted. If $W_L$ beams are accepted in a given layer, the remaining beams are discarded, and the method is moved on to verify the next layer. If fewer than $W_L$ beams are accepted in layer $l$, the method rejects this layer and terminates the verification process.

When verification ends, either because it is terminated or because there are no more layers to verify, our method samples an additional layer with $W_L$ beams. This additional layer either corrects the first rejected layer or adds a new layer if all draft layers are accepted. The output beams from this additional layer then serve as the input beams for the next iteration, continuing until the stopping criteria are met (e.g., reaching the maximum number of tokens).

This approach allows each run of the large model to produce at least one, and possibly multiple, steps of beam sampling. Previous studies have shown that memory operations during LLM runs contribute significantly to both runtime and energy consumption (Leviathan, Kalman, and Matias 2023; Allen and Ge 2016; Chen et al. 2011). By generating multiple tokens in a single run, DSBD reduces the number of memory operations required, which in turn improves both the speed and the energy efficiency of LLM inference.

**Details** Let $p$ denote the output distribution of the large model and $q$ denote the distribution of the small model. We will start by explaining how to verify the first draft layer

(which is the second layer of the draft forest) during each iteration.

Let $\mathcal{I} = \{x_{1:t}^{(1)}, \cdots, x_{1:t}^{(W_L)}\}$ represent the input beams, and $\mathcal{S} = \{x_{1:t+1}^{(1)}, \cdots, x_{1:t+1}^{(W_S)}\}$ represent the draft beams in the first layer of the draft forest. Note that $x_{1:t+1}^{(i)}$ is sampled from the distribution $q_{beam}(x_{1:t+1}^{(i)}) = \mathcal{T} \circ \frac{q(x_{1:t+1}^{(i)})}{\mathcal{Q}}$, where $\mathcal{T}$ denotes the warping operation and $\mathcal{Q} = \sum_{x_{1:t+1} \in \mathcal{I} \times V} q(x_{1:t+1})$. Similarly, let $p_{beam}$ denote the beam sampling distribution of the large model, we have $p_{beam}(x_{1:t+1}^{(i)}) = \mathcal{T} \circ \frac{p(x_{1:t+1}^{(i)})}{\mathcal{P}}$, where $\mathcal{P} = \sum_{x_{1:t+1} \in \mathcal{I} \times V} p(x_{1:t+1})$.

During verification, our method starts by setting $p' = p_{beam}$. For each draft beam $x_{1:t}^{(i)}$, DSBD accepts it with probability $\min(1, \frac{q_{beam}(x_{1:t}^{(i)})}{p'(x_{1:t}^{(i)})})$. If $x_{1:t}^{(i)}$ is rejected, the method updates $p'$ by setting it to $norm(\max(0, p' - q_{beam}))$, where $norm$ denotes the normalization operation. Then it continues to verify the next draft beam with the updated $p'$. If the beam is accepted, $p'$ is reset to $p_{beam}$. If $W_L$ draft beams have already been accepted in this layer, the method will reject all remaining beams.

Now we illustrate how to verify the second draft layer. The difference is that some beams in the first layer have already been rejected. In this case, all the beams stem from the rejected beams are directly rejected. For the remaining beams, DSBD applies the same verification process as above.

If all layers in the draft forest have $W_L$ accepted beams, the method proceeds to sample an additional layer with $W_L$ beams directly from $p_{beam}$. However, if at any layer $l$ fewer than $W_L$ beams are accepted, the method will first sample one beam from the adjusted distribution $p'$. If the number of accepted beams still falls short of $W_L$, additional beams will be sampled from the original distribution $p_{beam}$ to meet the required number.

**Theorem 3.1.** ***Correctness of Draft and Verification Scheme***. *Let $\mathcal{I} = \{x_{1:t}^{(1)}, \cdots, x_{1:t}^{(W_L)}\}$ denote input beams, $\mathcal{S} = \{x_{1:t+1}^{(1)}, \cdots, x_{1:t+1}^{(W_S)}\}$ denote draft beams, and $\mathcal{O} = \{\tilde{x}_{1:t+1}^{(1)}, \cdots, \tilde{x}_{1:t+1}^{(W_L)}\}$ denote the output beams obtained by our algorithm. We have $\tilde{x}_{1:t+1}^{(i)} \overset{\text{iid}}{\sim} p_{beam}$ ($\forall i = 1, \ldots, W_L$), where $p_{beam}(x_{1:t+1}^{(i)}) = \mathcal{T} \circ (p(x_{1:t+1}^{(i)})/\mathcal{P})$, $\mathcal{P} = \sum_{x_{1:t+1} \in \mathcal{I} \times V} p(x_{1:t+1})$.*

The proof is illustrated in (Qin et al. 2024).

## 3.2 Dynamic-Width Speculative Beam Decoding

The draft and verification scheme described above ensures that our method matches the sampling distribution of beam sampling. However, it has a limitation: the beam width $W_L$ remains fixed across all layers. While this fixed width works well for standard beam sampling, it is not suitable for our method. The key challenge is that the discrepancy between the small model's predictions ($q_{beam}$) and the large model's true distribution ($p_{beam}$) vary from token to token. In some layers, $q_{beam}$ closely aligns with $p_{beam}$, resulting in a high acceptance rate. In other layers, the gap is much wider, leading to a lower acceptance rate.

To address this variability, the decoding algorithm should dynamically adjust the number of beams it expects to accept based on the alignment between $q_{beam}$ and $p_{beam}$. By doing so, it can (1) reduce the target width for challenging layers, preventing the entire layer from being rejected and thus maintaining progress, and (2) increase the target width for

Algorithm 1: Draft and Verification for Speculative Beam Sampling

**Input:** Draft Forest with $\gamma$ draft layers, Small model distribution $q$, Large model distribution $p$, Beam width $W_L, W_S$.
**Output:** Verified beams for the next iteration
$l_{last} \leftarrow \gamma + 1$
**for** $l = 1, \ldots, \gamma$ **do**
  // $\mathcal{I}^{(l)}$ is the beams of layer $l-1$ in the forest.
  $\mathcal{I}^{(l)} \leftarrow$input beams of layer $l$.
  // $\mathcal{S}^{(l)}$ is the beams of layer $l$ in the forest.
  $\mathcal{S}^{(l)} \leftarrow$draft beams of layer $l$.
  // remove beams stem from beams rejected in the last layer
  $\mathcal{S}^{(l)} \leftarrow \{x_{1:t+1}^{(l,i)} | x_{1:t+1}^{(l,i)} \in \mathcal{S}^{(l)}, x_{1:t}^{(l,i)}$ is not rejected$\}$
  // $t+1$ is the length of sequence in $\mathcal{S}^{(l)}$, $t = l - 1$.
  compute $p_{beam}^{(l)}$ based on next-token distributions $p$
  $p' \leftarrow p_{beam}^{(l)}$
  Compute $W_L^{(l)}$ based on Eq 2 - Eq. 6
  **for** $x_{1:t+1}^{(l,i)} \in \mathcal{S}^{(l)}$ **do**
    $r \leftarrow U(0,1)$
    **if** $r \leq \frac{q_{beam}^{(l)}(x_{1:t+1}^{(i)})}{p'(x_{1:t+1}^{(i)})}$ **then**
      accept $x_{1:t+1}^{(l,i)}$
      $p' \leftarrow p_{beam}^{(l)}$
    **else**
      reject $x_{1:t+1}^{(l,i)}$
      $p' \leftarrow \text{norm}(\max(0, p' - q_{beam}^{(l)}))$
    **if** $W_L^{(l)}$ beams are accepted **then**
      reject remaining beams
      **break**
  **if** less than $W_L^{(l)}$ beams are accepted **then**
    sample $x_{1:t+1} \sim p'$ and add it to accepted beams
    **while** less than $W_L^{(l)}$ beams are accepted **do**
      sample $x_{1:t+1} \sim p_{beam}^{(l)}$ and add it to accepted beams
    $l_{last} \leftarrow l$
    **break**
**if** $l_{last} == \gamma + 1$ **then**
  compute $p_{beam}^{(\gamma+1)}$
  sample $W_L$ beams from $p_{beam}^{(\gamma+1)}$

**return** accepted beams at the layer $l_{last}$

easier layers, enhancing the exploration of diverse sequences and reducing the risk of getting trapped in local optima. This adaptive approach would optimize the balance between efficiency and accuracy, making the decoding process more robust and effective. So we propose a self-adjusting method where the target width $W_L^{(l)}$ for layer $l$ is determined based on the context of that layer.

Let $P_{p,q}^{(l)}(m,k)$ represent the probability that $k$ out of $m$ draft beams are accepted at the $l$-th layer. This probability is computed using the following recursive equation:

$$P_{p,q}^{(l)}(m,k) = \sum_{i=1}^{m} \tilde{P}_{p,q}^{(l)}(m,i) P_{p,q}(m-i, k-1)) \quad (2)$$

Here, $\tilde{P}_{p,q}^{(l)}(m,i)$ is the probability that the $i$-th beam is the first to be accepted among the $m$ draft beams:

$$\tilde{P}_{p,q}^{(l)}(m,i) = \alpha_i^{(l)} \prod_{j=1}^{i-1} (1-\alpha_j^{(l)}) \quad (3)$$

where $\alpha_j^{(l)}$ is the probability that the $j$-th beam is accepted, given that all previous beams (from the 1st to the $(j-1)$-th) were rejected.

$$\alpha_j^{(l)} = \sum q_{beam} \min(p_j^{(l)}/q_{beam}^{(l)}, 1) \quad (4)$$

where $p_1^{(l)} = p_{beam}^{(l)}$, $p_k^{(l)} = norm(\max(p_{k-1}^{(l)} - q_{beam}^{(l)}, 0))$.

Using these equations and the fact that $P_{p,q}^{(l)}(m,k) = 0$ if $k > m$ and $P_{p,q}^{(l)}(0,0) = 1$, we can calculate the probability that at least $K$ beams are accepted at the $l$-th layer as:

$$1 - \sum_{k=1}^{K-1} P_{p,q}^{(l)}(M_S, k) \quad (5)$$

Finally, the width $W_L^{(l)}$ for the $l$-th layer is set based on Eq 5, ensuring that it is not less than a minimum width $W_{min}$:

$$W_L^{(l)} = \max(W_{min}, \tilde{W}_L^{(l)}(t)) \quad (6)$$

In this expression, $t \in [0,1]$ is a pre-defined threshold. The value of $\tilde{W}_L^{(l)}(t)$ is computed as follows:

$$\tilde{W}_L^{(l)}(t) = \max\{K \in \mathbb{N} | 1 - \sum_{k=0}^{K-1} P_{p,q}^{(l)}(M_S, k) \geq t\} \quad (7)$$

This formula gives us the maximum width $\tilde{W}_L^{(l)}(t)$ such that the probability of accepting at least $\tilde{W}_L^{(l)}(t)$ beams at the $l$-th layer is greater than or equal to the threshold $t$. Eq 6 ensures that the width is dynamically adjusted to maintain a high likelihood of accepting a sufficient number of beams, while also ensuring that it does not fall below the minimum width $Wmin$. Algorithm 1 illustrates the pseudocode for the draft and verification scheme.

Let $\beta_{W_{min}}^{(l)} = \sum_{k=W_{min}}^{W_S} P_{p,q}^{(l)}(W_S, k))$, which is the probability that at least $W_{min}$ beams are accepted at layer $l$. Based on the definition of $W_L^{(l)}$, the probability that layer $L$ is accepted is $\min(t, \beta_{W_{min}}^{(l)})$. So $t$ and $W_{min}$ both control the average acceptance rate of our algorithm, and hence determine efficiency. Let $\bar{\beta} = \mathbb{E}\beta_{W_{min}}^{(l)}$, we have the following theorem for the efficiency of DSBD.

**Theorem 3.2.** *The expected number of steps generated per iteration is* $\frac{1-\min(t,\beta)^{\gamma+1}}{1-\min(t,\beta)}$.

*Proof.* As described above, the average acceptance rate is $\min(t,\beta)$. With the Theorems in (Leviathan, Kalman, and Matias 2023), we can calculate the average number of generated layers as $\frac{1-\min(t,\beta)^{\gamma+1}}{1-\min(t,\beta)}$. □

### 3.3 Forest-based Parallel Decoding

As noted in (Miao et al. 2023), efficient management of the key-value cache is crucial to avoid redundant computations when running the large model during verification, which affects overall efficiency. SpecInfer (Miao et al. 2023) introduced tree-based parallel decoding, which reuses the same key-value cache and employs a topology-aware causal mask to accelerate the computation of the large model. However, this tree-based parallel decoding approach cannot be directly applied to our algorithm because, unlike SpecInfer, our method retains multiple beams as inputs at each iteration. Although these beams share the same initial input, the tokens generated in each beam can differ significantly as the sequence length increases. As a result, the draft tokens in DSBD form not a single tree but a forest.

So we propose forest-based parallel decoding, an extension of tree-based parallel decoding that accommodates multiple trees. As shown in Figure 3, DSBD maintains a separate key-value cache for each input beam. For each beam, we apply tree-based parallel decoding to compute the tree attention across all tokens. Finally, after the iteration ends, DSBD updates the key-value caches according to the output beams. For example, if the output beams in Figure 3 are $b_5$ and $b_6$, which both originate from $b_1$, then the caches for $b_5$ and $b_6$ are kept for the next iteration.

### 3.4 Reducing Memory Cost

In practice, key-value caches take up a large portion of memory cost for LLM inference (Kang et al. 2024). A critical disadvantage of beam sampling is that it has to maintain a separate key-value cache for each beam, significantly increasing the memory cost. But our method can mitigate this issue with a simple modification. Notice that with the forest-based parallel decoding, the number of key-value caches kept during generation equals the number of input beams. So an effective way to reduce the memory cost of our method is to limit the number of input beams. This can be achieved by selecting only the output beam with the lowest perplexity as the input beam for the next iteration. In this way, only one key-value cache is needed during generation, so the memory cost will be similar to the cost of multinomial sampling and speculative decoding. Notice that although there is only one input beam, more than one beam can be accepted at each layer of the draft forest. Hence, it will be more effective than multinomial sampling.

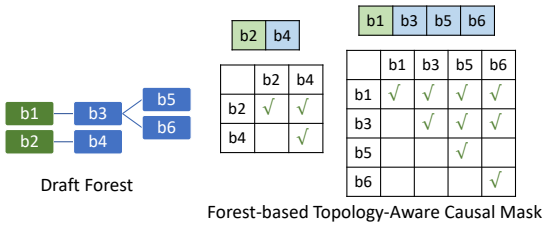



Figure 3: Illustration of forest-based parallel decoding. Given the draft forest, the large model converts the two trees into sequences in depth-first search order and verifies them in parallel with the topology-aware attention mask. Empty cells in the matrices indicate that attention is masked.

# 4 Experiments

## 4.1 Experiment Setups

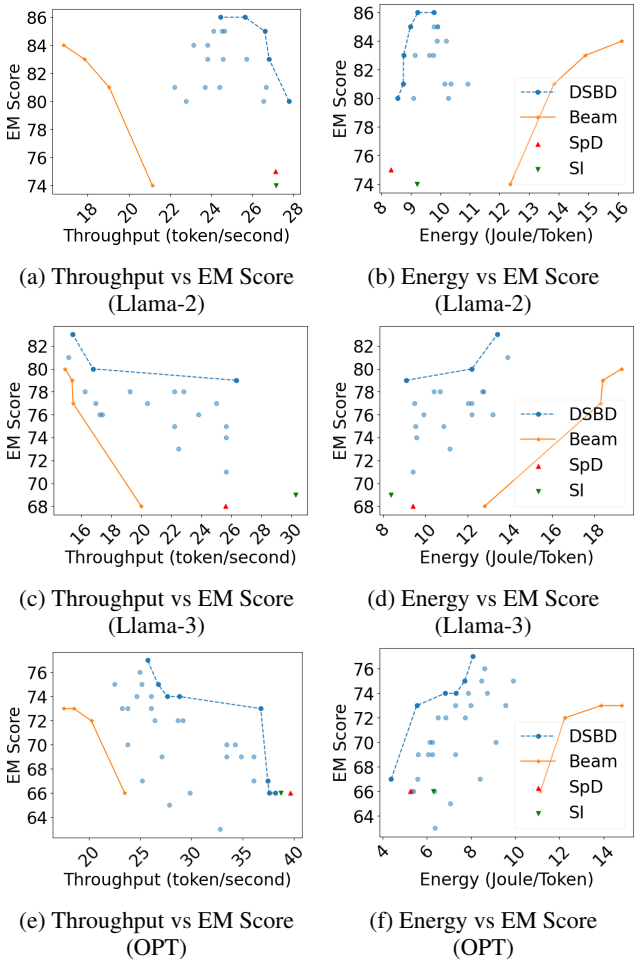
**LLMs**. We evaluate our method using three publicly available LLM families: OPT (Zhang et al. 2022), Llama-2 and Llama-3 (Touvron et al. 2023; AI@Meta 2024). We use Llama-2-13B, Llama-3.1-8B, and OPT-13B as the large models as they are the largest models our GPU could run. And we use Llama-68M (Miao et al. 2023) , Llama-3.2-1B, and OPT-125M as the small models.

**Datasets**. We use public datasets: SQuAD (Rajpurkar, Jia, and Liang 2018), Spider (Yu et al. 2018), and MT-Bench (Zheng et al. 2023). SQuAD is a natural language QA dataset using exact match (EM) as the evaluation metric. Spider is a text-to-SQL code dataset that uses execution accuracy (EA) as the metric. MT-bench covers various tasks including writing, roleplay, extraction, stem, humanities, reasoning, math, and coding. It uses GPT-4 to rate the output quality on a scale of 1-10 (the higher the better).[2]

## 4.2 Comparison with Beam Sampling

We begin by comparing DSBD with beam sampling, focusing on the relationship between efficiency (e.g., energy consumption and throughput) and effectiveness. The width of beam sampling ranges from 1 to 4. When width equals 1, beam sampling is equivalent to multinomial sampling. In addition, we observe the improvement in downstream effectiveness and output perplexity begins to converge when the width reaches around 4. For our method, we vary the draft beam width $W_S \in \{2, 3, 4, 5, 6\}$, the threshold $t \in \{0.7, 0.9\}$, and set $W_{min} \in \{1, 2, 3\}$. We also include speculative decoding (Leviathan, Kalman, and Matias 2023) (SpD) and SpecInfer (Miao et al. 2023) (SI) as references.

Figure 4 and Figure 5 illustrate the points that mark the performance of different methods under different parameter settings on SQUAD and Spider datasets, respectively. SpD and SpecInfer each have only one point in the figures because they do not offer a trade-off between efficiency and effectiveness. We plot the curves of beam sampling and the Pareto fronts of DSBD. Notably, we omit the results of the OPT model on the Spider dataset as its execution accuracy remains consistently close to zero, rendering it uninformative for this analysis. The figures demonstrate that DSBD consistently outperforms beam sampling, signifying that it achieves higher quality at any given level of throughput or energy consumption. More importantly, when the effectiveness is fixed, DSBD can be 1.5-1.9× faster than beam sampling, while reducing energy consumption by 1.8-2.5×, as demonstrated by the Pareto fronts of DSBD. Table 1 presents the results on MT-Bench. Due to the cost and time of GPT-4 evaluations, we report results for SpecInfer, beam sampling ($W = 5$), and DSBD. DSBD achieves comparable efficiency to SpecInfer while significantly improving output quality. It is also 1.53× faster and 1.54× more energy-efficient than beam sampling. These results highlight DSBD's advantages in efficiency and effectiveness, making it ideal for real-world applications.



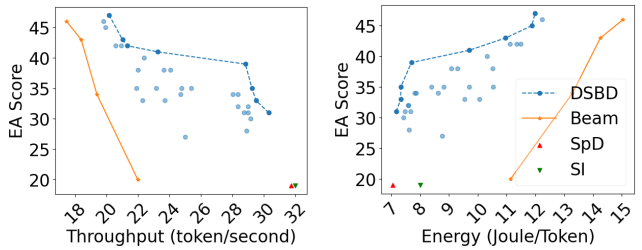

(a) Throughput vs EM Score (Llama-2)
(b) Energy vs EM Score (Llama-2)
(c) Throughput vs EM Score (Llama-3)
(d) Energy vs EM Score (Llama-3)
(e) Throughput vs EM Score (OPT)
(f) Energy vs EM Score (OPT)

Figure 4: Evaluation on SQuAD. Exact match (EM) is **higher** the better. The blue points represent performances of DSBD under different parameter settings $(\gamma, W_S, t)$. The blue and yellow lines mark the Pareto fronts of DSBD and beam sampling. (SpD: SpecDecode, SI: SpecInfer)

## 4.3 Comparison under Memory Constraint

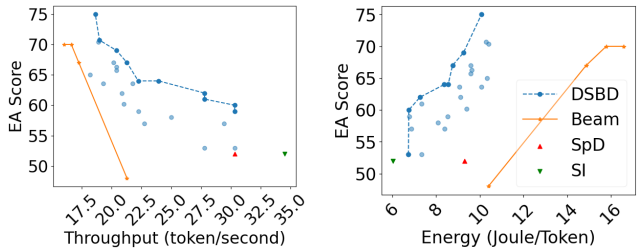
As discussed in Section 3.4, DSBD can mitigate the memory issue of beam sampling by selecting only one output

[2]Additional experiments and reproduction details are available in our arXiv version (Qin et al. 2024).




(a) Speed vs EA Score (Llama-2)

(b) Energy vs EA Score (Llama-2)




(c) Speed vs EA Score (Llama-3)

(d) Energy vs EA Score (Llama-3)

Figure 5: Evaluation on Spider. Execution accuracy (EA) is **higher** the better. The blue points represent performances of DSBD under different parameter settings $(\gamma, W_S, t)$. The blue and yellow lines mark the Pareto fronts of DSBD and beam sampling. (SpD: SpecDecode, SI: SpecInfer)

| Model | Method | Score | Token/s | J/token |
|---|---|---|---|---|
| Llama-2-13B | SpecInfer | 2.86 | **21.8** | 21.2 |
| | Beam ($W$=5) | 3.51 | 12.1 | 26.3 |
| | DSBD | **3.52** | 16.5 | **16.1** |
| Llama-3-8B | SpecInfer | 3.46 | **20.2** | **19.8** |
| | Beam ($W$=5) | 4.10 | 10.5 | 33.3 |
| | DSBD | **4.11** | 17.8 | 22.9 |

Table 1: Evaluation on MT-Bench with SpecInfer, beam sampling and DSBD.

beam for the next iteration. It allows DSBD to only keep one sequence of key-value cache and to achieve memory usage comparable to that of multinomial sampling. To assess the performance of DSBD under memory constraints (i.e., only keeps one sequence of key-value cache), we compare it with multinomial sampling, SpD, and SpecInfer, as shown in Table 2. In addition, the DSBD in Table 1 also only keeps one sequence of key-value cache. The results show that DSBD achieves speed and energy efficiency close to that of SpD. Moreover, DSBD delivers a significant improvement in output quality, far surpassing the baselines in downstream scores.

## 5 Related Work

**Efficient LLM Inference**. Numerous studies have focused on improving the efficiency of large model inference, including model quantization (Frantar et al. 2022; Lin et al. 2023), model pruning (Gale, Elsen, and Hooker 2019; Sanh, Wolf, and Rush 2020), and model distillation (Hinton, Vinyals, and Dean 2015). While these techniques achieve significant speed-ups, they often sacrifice the model's overall effectiveness. A closely related direction to our work is non-autoregressive decoding, enabling parallel generation of multiple tokens (Gu et al. 2017; Wang et al. 2019; Sun et al. 2019; Ghazvininejad et al. 2019; Lee, Mansimov, and Cho 2018; Guo, Xu, and Chen 2020). However, these methods typically require extensive retraining of the model and often face challenges in either maintaining model effectiveness or achieving comparable performance without relying on task-specific techniques (Kim et al. 2023).

| | Method | EM/EA | tokens/s | J/token |
|---|---|---|---|---|
| Llama-2 SQuAD | Multinomial | 74 | 21.14 | 12.36 |
| | SpD | 75 | 27.11 | **8.34** |
| | SpecInfer | 74 | **27.15** | 9.22 |
| | DSBD | **86** | 26.67 | 8.75 |
| Llama-2 Spider | Multinomial | 20 | 21.98 | 11.14 |
| | SpD | 19 | 31.74 | **7.06** |
| | SpecInfer | 19 | **32.00** | 8.01 |
| | DSBD | **31** | 30.30 | 7.17 |

Table 2: Comparison under memory constraints: each method stores key-value caches for only one sequence.

**Speculative Decoding**. Speculative decoding is initially introduced in (Leviathan, Kalman, and Matias 2023; Chen et al. 2023). More recent works (Sun et al. 2023; Miao et al. 2023; Yang et al. 2024) extend this concept by allowing the small model to generate multiple draft sequences. All these methods only maintains a single sequence during generation, making them prone to sub-optimal results. Recently, Andronov et al. (Andronov et al. 2024) proposed a decoding method called "speculative beam search". While it retains multiple candidate sequences to handle the chemical synthesis planning task, it does not preserve the same distribution as either beam sampling or multinomial sampling, and their method is fundamentally different from ours. Another complementary direction to enhance speculative decoding is to improve the effectiveness of the small draft model. A more effective draft model leads to a higher acceptance rate of draft tokens, which in turn accelerates the overall inference process (Kim et al. 2023; Liu et al. 2023; He et al. 2023). EAGLE (Li et al. 2024) and MEDUSA (Cai et al. 2024) train additional heads in the target model to generate draft tokens and achieve better acceptance rate. These works are orthogonal to our work because our algorithm can be directly applied to their draft models.

## 6 Conclusion

This work introduces a novel method that integrates speculative decoding with beam sampling to enhance the efficiency and effectiveness of large language model (LLM) inference. Experimental results show that DSBD outperforms beam sampling, achieving a significant speed-up and energy reduction without compromising downstream task performance. This work enhances the effectiveness of speculative decoding and opens new avenues for exploration.

## Acknowledgements

This work was partially supported by NSF grants 2211557, 1937599, 2119643, 2303037, NSF 2312501, SRC JUMP 2.0 PRISM Center, NASA, Okawa Foundation, Amazon Research, Snapchat, and the CDSC industrial partners (https://cdsc.ucla.edu/partners/).

## References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 Model Card.

Allen, T.; and Ge, R. 2016. Characterizing power and performance of gpu memory access. In *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*, 46–53. IEEE.

Andronov, M.; Andronova, N.; Wand, M.; Schmidhuber, J.; and Clevert, D.-A. 2024. Accelerating the inference of string generation-based chemical reaction models for industrial applications. *arXiv preprint arXiv:2407.09685*.

Anil, R.; Dai, A. M.; Firat, O.; Johnson, M.; Lepikhin, D.; Passos, A.; Shakeri, S.; Taropa, E.; Bailey, P.; Chen, Z.; et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Cai, T.; Li, Y.; Geng, Z.; Peng, H.; Lee, J. D.; Chen, D.; and Dao, T. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

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

Chen, J.; Li, B.; Zhang, Y.; Peng, L.; and Peir, J.-k. 2011. Tree structured analysis on GPU power study. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, 57–64. IEEE.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Ghazvininejad, M.; Levy, O.; Liu, Y.; and Zettlemoyer, L. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Gu, J.; Bradbury, J.; Xiong, C.; Li, V. O.; and Socher, R. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Guo, J.; Xu, L.; and Chen, E. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 376–385.

He, Z.; Zhong, Z.; Cai, T.; Lee, J. D.; and He, D. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Kang, H.; Zhang, Q.; Kundu, S.; Jeong, G.; Liu, Z.; Krishna, T.; and Zhao, T. 2024. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*.

Kim, S.; Mangalam, K.; Moon, S.; Malik, J.; Mahoney, M. W.; Gholami, A.; and Keutzer, K. 2023. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lee, J.; Mansimov, E.; and Cho, K. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Leviathan, Y.; Kalman, M.; and Matias, Y. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 19274–19286. PMLR.

Li, Y.; Wei, F.; Zhang, C.; and Zhang, H. 2024. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees. *arXiv preprint arXiv:2406.16858*.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Dang, X.; and Han, S. 2023. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978*.

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

Miao, X.; Oliaro, G.; Zhang, Z.; Cheng, X.; Wang, Z.; Wong, R. Y. Y.; Chen, Z.; Arfeen, D.; Abhyankar, R.; and Jia, Z. 2023. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 1(2): 4.

Qin, Z.; He, Z.; Prakriya, N.; Cong, J.; and Sun, Y. 2024. Dynamic-Width Speculative Beam Decoding for Efficient LLM Inference. *arXiv preprint arXiv:2409.16560*.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822*.

Sanh, V.; Wolf, T.; and Rush, A. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33: 20378–20389.

Shi, C.; Yang, H.; Cai, D.; Zhang, Z.; Wang, Y.; Yang, Y.; and Lam, W. 2024. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925*.

Sun, Z.; Li, Z.; Wang, H.; He, D.; Lin, Z.; and Deng, Z. 2019. Fast structured decoding for sequence models. *Advances in Neural Information Processing Systems*, 32.

Sun, Z.; Suresh, A. T.; Ro, J. H.; Beirami, A.; Jain, H.; and Yu, F. 2023. Spectr: Fast speculative decoding via optimal transport. *arXiv preprint arXiv:2310.15141*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Y.; Tian, F.; He, D.; Qin, T.; Zhai, C.; and Liu, T.-Y. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5377–5384.

Yang, S.; Huang, S.; Dai, X.; and Chen, J. 2024. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*.

Yang, Z.; Adamek, K.; and Armour, W. 2023. Part-time Power Measurements: nvidia-smi's Lack of Attention. *arXiv preprint arXiv:2312.02741*.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.