DynaTree: Dynamic Tree-based Speculative Decoding with Adaptive Pruning for Efficient LLM Inference

Nuoyan Chen[*] **Jiamin Liu**[*] **Zhaocheng Li**[*]
School of Computer Science
Shanghai Jiao Tong University
`{cny123222, logic-1.0, lzc050419}@sjtu.edu.cn`

Abstract

Autoregressive decoding in large language models (LLMs) is fundamentally sequential and therefore underutilizes modern accelerator parallelism during token generation. Speculative decoding mitigates this bottleneck by letting a lightweight draft model propose multiple tokens that are verified in parallel by the target model; however, common linear variants explore only a single draft chain per step and can waste substantial computation when early tokens are rejected. We propose **DynaTree**, a tree-based speculative decoding framework that drafts multiple candidate continuations via top-k branching and verifies the resulting token tree in one forward pass using tree attention. To control the exponential growth of the draft tree, DynaTree applies adaptive pruning that removes low-probability branches under an explicit node budget. Experiments on Pythia models show that DynaTree improves decoding throughput by up to $1.62\times$ over standard autoregressive generation and consistently outperforms strong speculative decoding baselines across generation lengths.

1 Introduction

Large language models (LLMs) are typically deployed with autoregressive decoding, where each output token is generated after conditioning on all previously generated tokens. While transformer inference can exploit parallelism during the prefill stage, the decode stage remains inherently sequential and requires a full forward pass per token, leading to poor hardware utilization and high latency [8, 5, 2].

Speculative decoding alleviates this bottleneck by separating *proposal* and *verification* [4]. A small draft model proposes several candidate tokens, and the target model verifies them in parallel; when the proposal matches the target distribution, multiple tokens can be committed per iteration. Importantly, with rejection sampling, speculative decoding preserves the exact output distribution of the target model [7].

In practice, most speculative decoding systems employ *linear* drafting: the draft model proposes a single chain of K tokens. This design is brittle under draft–target mismatch: a rejection at an early position forces all subsequent drafted tokens to be discarded, wasting both draft computation and target-model verification work, and constraining achievable speedups [12].

We argue that the single-path constraint is unnecessary. When multiple plausible next tokens compete, exploring several continuations in parallel increases the chance that at least one path aligns with the target model, thereby improving the expected number of accepted tokens per verification step. This motivates *tree-based* speculation, where the draft expands multiple candidates via top-k branching and the target verifies the resulting token tree with a structured, causality-preserving attention mask.

[*]Equal contribution.

The central challenge is controlling verification cost: naive tree expansion grows exponentially with depth and branching. We present **DynaTree**, a tree-based speculative decoding framework with a lightweight adaptive pruning mechanism that removes low-probability branches while enforcing an explicit node budget. Empirically, DynaTree achieves up to $1.62\times$ throughput improvement over standard autoregressive decoding on Pythia models and consistently outperforms strong speculative decoding baselines. In summary, our contributions are: (i) a practical tree-based speculative decoding algorithm with efficient tree attention verification; (ii) an adaptive pruning strategy that stabilizes the depth–breadth trade-off under a fixed verification budget; and (iii) an extensive empirical study characterizing these trade-offs across generation lengths.

2 Related Work

2.1 Speculative Decoding

Speculative decoding accelerates autoregressive generation by decoupling *proposal* and *verification*: a lightweight draft model proposes multiple tokens, and the target model verifies these candidates in parallel while preserving the exact output distribution through rejection sampling [4, 7]. Empirical and theoretical analyses highlight that achievable speedups depend critically on the acceptance behavior induced by the draft–target mismatch, and on the additional overhead introduced by drafting and verification [12, 10]. Despite strong progress, the dominant implementation remains *linear* drafting, where a single speculative chain is proposed per iteration; when early tokens are rejected, downstream drafted tokens are discarded, causing substantial wasted computation and limiting utilization of parallel verification.

2.2 Tree-Based and Parallel Decoding

To overcome the single-path limitation, recent work explores *tree-based* speculative decoding, where multiple candidate continuations are drafted and verified in a single target-model forward pass using structured attention masks. SpecInfer [6] instantiates this idea in an LLM serving setting by building a token tree and verifying it efficiently. OPT-Tree [9] further studies *adaptive* tree construction, selecting tree shapes to maximize expected acceptance length under a fixed verification budget. In parallel, Medusa [1] pursues multi-token generation by augmenting a base model with multiple decoding heads and verifying the induced candidate tree; unlike draft–target speculative decoding, it requires model-specific fine-tuning. Our work follows the draft–target paradigm but focuses on practical tree construction and verification under strict budget constraints, emphasizing dynamic pruning as a lightweight mechanism to stabilize performance.

2.3 Dynamic Pruning Strategies

Tree-based methods must contend with the exponential growth of candidates with depth and branching. ProPD [13] proposes dynamic token-tree pruning and generation, leveraging early signals to remove low-utility branches before full verification. Cost-aware formulations further model verification overhead and explicitly optimize the trade-off between exploration and target-model computation [3]. DySpec [11] employs greedy, confidence-guided expansion to adapt tree structures online. DynaTree is closely related to these approaches: we adopt a probability-threshold-based pruning rule coupled with an explicit node budget, aiming for a simple, training-free mechanism that is easy to integrate while maintaining strong speedups in practice.

3 Methodology

3.1 Problem Setup and Notation

Let M_T denote a target autoregressive language model and M_D a smaller draft model. Given a prefix (prompt) $x_{1:t}$, greedy decoding with M_T produces tokens $y_{t+1}, y_{t+2}, \ldots$ where

$$y_i \;=\; \arg\max_{v\in\mathcal{V}} p_T(v \mid x_{1:i-1}).$$

Figure 1: **Architecture overview (placeholder).** DynaTree iteration pipeline: draft-tree construction (with pruning), BFS flattening, tree-attention verification in one target forward pass, and commit/cache update for the longest valid path.

Pseudocode (one DynaTree iteration, greedy decoding).

```
1. Draft a token tree $\mathcal{T}$ with parameters $(D, B, \tau, N_{\max})$.
2. Flatten $\mathcal{T}$ in BFS order to obtain tokens $z_{1:n}$.
3. Build the tree attention mask $\mathbf{A}$ (prefix + ancestors only).
4. Run one target forward pass on $z_{1:n}$ with $\mathbf{A}$ to get logits.
5. Select the longest root-to-leaf path whose tokens match target greedy predictions.
6. Commit the matched tokens and append one target "bonus" token.
7. Crop target KV cache back to the pre-iteration prefix and rebuild it on committed tokens.
```

Figure 2: **DynaTree iteration pseudocode.** (Rendered without algorithm packages for portability.)

Speculative decoding accelerates generation by proposing candidate tokens with M_D and verifying them with M_T, while preserving the greedy output when the verification rule only commits tokens that match the target greedy predictions.

3.2 Overview of DynaTree

DynaTree generalizes linear speculative decoding from a single draft chain to a *draft token tree*. In each iteration, DynaTree performs: (i) **tree drafting** with M_D by expanding top-B candidates up to depth D; (ii) **parallel verification** of all drafted nodes using a tree attention mask in a single forward pass of M_T; (iii) **path selection and commit** by greedily selecting the longest path consistent with the target model's greedy predictions; and (iv) **KV-cache update** for the committed tokens.

3.3 Draft Tree Construction with Dynamic Pruning

We maintain a token tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ whose nodes $u \in \mathcal{N}$ correspond to drafted tokens. Each node stores: token $z_u \in \mathcal{V}$, parent $\pi(u)$, depth $d(u)$, draft log-probability $\ell_u = \log p_D(z_u \mid \text{prefix}(\pi(u)))$, and cumulative log-probability $\bar{\ell}_u = \sum_{v \in \text{path}(u)} \ell_v$.

Expansion. Starting from the current prefix $x_{1:t}$, we first obtain the draft distribution $p_D(\cdot \mid x_{1:t})$ and create the root node u_0 using the top-1 token. Then, for each active leaf u with $d(u) < D$, we expand children by selecting the top-B tokens under $p_D(\cdot \mid x_{1:t+\text{pos}(u)})$ (implemented via cached one-token forward passes). To bound computation, we enforce a hard **node budget** $N_{\max}$ and stop expansion when $|\mathcal{N}| \geq N_{\max}$.

Probability-threshold pruning (V2). To avoid wasting draft computation on unlikely branches, DynaTree prunes any leaf u whose cumulative probability falls below a threshold $\tau \in (0, 1)$:

$$\bar{\ell}_u \; < \; \log \tau \quad \Rightarrow \quad \text{prune } u.$$

This rule corresponds exactly to the implementation in 'TreeSpeculativeGeneratorV2', where $\bar{\ell}_u$ is accumulated along the path and compared against $\log \tau$.

Figure 3: **Tree attention mask (placeholder).** Example tree, BFS flattening, and the induced causal mask where each node attends only to the prefix and its ancestors.

3.4 Tree Attention for Parallel Verification

To verify all drafted tokens in one target-model forward pass, we *flatten* the tree in breadth-first order (BFS), producing a sequence $z_{1:n}$ where each token corresponds to one node and all ancestors appear earlier than descendants. We then construct a boolean attention mask $\mathbf{A} \in \{0,1\}^{n\times(t+n)}$ such that each drafted token attends to: (i) all prefix tokens $x_{1:t}$, and (ii) only its ancestors (including itself) in the flattened tree:

$$\mathbf{A}_{i,j} = \begin{cases} 1, & 1 \le j \le t, \\ 1, & j = t + \mathrm{pos}(v) \text{ for some ancestor } v \in \mathrm{Anc}(u_i) \cup \{u_i\}, \\ 0, & \text{otherwise.} \end{cases}$$

This mask ensures the conditional distribution computed at each node matches the distribution of sequential decoding along its unique root-to-node path, while enabling parallel verification across different branches [6, 9].

3.5 Greedy Path Selection and Cache Update

Verification signals. Let $\hat{y}_{t+1} = \arg\max p_T(\cdot \mid x_{1:t})$ be the target model's greedy next token from the prefix (available from the prefix logits). For each tree node u with flattened position i, the target forward pass outputs logits $\mathbf{s}_i$, whose argmax $\hat{y}(u) = \arg\max \mathbf{s}_i$ corresponds to the greedy *next-token* prediction after consuming the path to u.

Longest valid path. DynaTree commits the longest path $u_0 \to u_1 \to \cdots \to u_m$ such that the drafted token at each node matches the target greedy prediction from its parent context:

$$z_{u_0} = \hat{y}_{t+1}, \quad z_{u_k} = \hat{y}(u_{k-1}) \ \text{ for } k = 1, \ldots, m.$$

If no drafted token matches the first greedy prediction, we fall back to committing $\hat{y}_{t+1}$ (one token progress). After committing the matched draft tokens, we append one *bonus* token $\hat{y}(u_m)$ from the target model, mirroring the greedy speculative decoding convention and ensuring steady progress.

KV-cache management. Tree verification may populate KV states for non-committed branches. To maintain correctness, we crop the target cache back to the pre-iteration prefix length t and then forward only the committed tokens to rebuild the cache (as implemented in ‘$_{u}pdate_{t}ree_{c}ache$‘).

3.6 Correctness for Greedy Decoding

We sketch the correctness argument for greedy decoding (the setting used throughout our experiments). The tree attention mask guarantees that for any node u, the target logits at u are computed from exactly the same conditioning context as in sequential decoding along the root-to-u path. DynaTree commits a drafted token *only if* it equals the target greedy argmax under that context. Therefore, every committed token matches the token that greedy decoding with M_T would produce at that position. The cache rollback-and-rebuild step ensures the subsequent iteration starts from an identical KV state. Consequently, DynaTree generates exactly the same token sequence as greedy decoding with the target model, while reducing the number of expensive target-model forward passes by verifying many candidate tokens in parallel.

Table 1: Main decoding throughput on Pythia (500-token generation). Higher is better.

Method	Throughput (t/s)	Speedup	Notes
AR (target-only)	119.4	1.00	Greedy decoding
HuggingFace assisted	161.9	1.36	`assistant_model`
Linear speculative (K=6)	133.1	1.11	Best linear in report
StreamingLLM + speculative (cache=1024)	132.9	1.11	Cache compression
DynaTree (Tree V2)	**193.4**	**1.62**	D=8, B=3, τ=0.03, $N_{\max}$=128

3.7 Complexity Discussion

Let $n = |\mathcal{N}| \leq N_{\max}$ be the number of drafted nodes. Drafting requires $O(n)$ one-token forward passes of the draft model (with cache reuse across expansions). Verification requires a single target-model forward pass over n tokens with a structured attention mask. Dynamic pruning reduces n in uncertain regions by discarding low-probability branches, improving the trade-off between draft overhead and verification parallelism.

4 Experiments

4.1 Experimental Setup

Models. We evaluate on the Pythia family with a target model M_T = Pythia-2.8B and a draft model M_D = Pythia-70M, using greedy decoding throughout (`do_sample=False`).

Hardware and software. All experiments are run on a single NVIDIA GPU. We use PyTorch 2.x and HuggingFace Transformers 4.x with `DynamicCache` for KV management.

Workloads. Unless otherwise stated, we generate 500 new tokens from a fixed technical prompt and report averages over 5 runs, skipping the first run as warmup (see ‘papers/Tree$_{speculative_Decoding.md}$‘).

4.2 Metrics

We report **throughput** (tokens/sec) as the primary metric. We also report **speedup** relative to the autoregressive baseline, and (when available) **acceptance rate** and **tokens per iteration** (average committed tokens per verification step).

4.3 Baselines

We compare against: (i) **AR** greedy decoding with the target model; (ii) **HuggingFace assisted generation** (built-in speculative decoding using `assistant_model`); (iii) **linear speculative decoding** implemented with a draft chain of length K; and (iv) **StreamingLLM + speculative decoding** for long-context cache compression.

4.4 Main Results

Table 1 summarizes end-to-end throughput on 500-token generation. DynaTree (Tree V2) achieves the best speedup (1.62$\times$) and outperforms both HuggingFace assisted generation (1.36$\times$) and the best linear speculative decoding setting (1.11$\times$).

4.5 Ablation Study: Progressive Component Addition

To isolate the contribution of tree structure and pruning/optimization, we use the exhaustive parameter sweep to form a progressive ablation sequence (Linear $\rightarrow$ Tree $\rightarrow$ Optimized Tree). Table 2 reports throughput and speedup under the sweep harness; while absolute numbers can differ from the end-to-end benchmark, the relative trend is consistent.

Figure 4: **Main results visualization (placeholder).** Bar chart of throughput/speedup for Table 1.

Table 2: Ablation study (progressive component addition) extracted from the sweep results (‘results/ablation$_{p}roper.json$‘).

Method	Configuration	Throughput (t/s)	Speedup
Linear speculative	K=6	133.1	1.11
+ Tree structure	D=4, B=3, τ=0.01	176.6	1.43
+ Depth & pruning optimization	D=8, B=3, τ=0.03	**221.4**	**1.79**

4.6 Hyperparameter Sweep

We perform a grid search over tree depth $D \in \{3,4,5,6,7,8\}$, branching factor $B \in \{2,3,4\}$, and pruning threshold $\tau \in \{0.01,0.02,0.03,0.05,0.1\}$ across multiple generation lengths (100–1000 tokens), totaling 450 configurations (‘results/tree$_{p}aram_{s}earch_{2}0251231_{1}40952.json$‘).$Figure\ 5willpresentthemaintrade-offs: increasingD$ and B improves exploration but raises verification cost, while τ controls the effective tree size by pruning low-probability branches.

4.7 Sequence Length Scaling

Table 3 reports the best-performing Tree V2 configuration selected per generation length, together with baseline throughput and speedup (extracted from the sweep; ‘results/length$_{s}caling_{e}xtracted.json$‘).$Weobservethatspeedupincreasesfromshorttomediumlengthsasverificationov$

5 Citations, figures, tables, references

These instructions apply to everyone.

5.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

```
http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf
```

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated...

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2025` package:

Figure 5: **Parameter sweep (placeholder).** Suggested 6-panel plot: speedup vs depth, vs branching factor, vs threshold; speedup vs length; average tree size heatmap; acceptance/tokens-per-round distribution.

Table 3: Best Tree V2 performance across different generation lengths (from ‘results/length$_{s}caling_{e}xtracted.json$‘).

Length	Optimal (D, B, τ)	Baseline (t/s)	Tree V2 (t/s)	Speedup	Accept.
100	(7,3,0.03)	105.3	150.7	1.43	0.31
200	(7,3,0.03)	125.5	193.2	1.54	0.36
300	(7,3,0.03)	124.2	199.0	1.60	0.38
500	(8,3,0.03)	123.9	221.4	1.79	0.38
1000	(6,3,0.05)	124.5	212.3	1.71	0.37

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2025}
```

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous" and include a copy of the anonymized paper in the supplementary material.

5.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[3]

5.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

[2]Sample of the first footnote.
[3]As in this example.

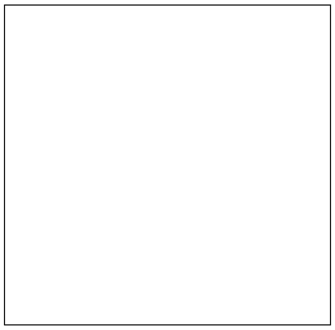

Figure 6: Sample figure caption.

Table 4: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

5.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 4.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 4.

5.5 Math

Note that display math in bare TeX commands will not create correct line numbers for submission. Please use LaTeX (or AMSTeX) commands for unnumbered display math. (You really shouldn't be using $$ anyway; see `https://tex.stackexchange.com/questions/503/why-is-preferable-to` and `https://tex.stackexchange.com/questions/40492/what-are-the-differences-between-align-equation-and-displaymath` for more information.)

5.6 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

6 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

6.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding (financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2025/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to automatically hide this section in the anonymized submission.

References

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

Transformer inference: Techniques for faster AI models. (2024). *PremAI Technical Report*. `https://www.premai.io/blog/transformer-inference-techniques-for-faster-ai-models`

Decoding real-time LLM inference: A guide to the latency vs throughput bottleneck. (2024). *Medium Technical Blog*.

Mitra, S. (2025). Making LLMs faster: My deep dive into speculative decoding. *Technical Blog Post*.

NVIDIA Developer. (2024). An introduction to speculative decoding for reducing latency in AI inference. *NVIDIA Technical Blog*.

Mitchell, E., Lee, J., Khazatsky, A., Manning, C. D., and Finn, C. (2024). Decoding speculative decoding. In *NAACL*.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Wong, R. Y. Y., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. (2024). SpecInfer: Accelerating generative large language model serving with tree-based speculative inference. In *ASPLOS*.

OPT-Tree: Speculative decoding with adaptive draft tree structure. (2025). *Transactions of the Association for Computational Linguistics (TACL)*.

Cai, T., Li, Y., Geng, Z., Peng, H., and Dao, T. (2024). Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *ICML*.

Wang, J., Xu, Y., Sun, L., and Wang, D. (2024). ProPD: Dynamic token tree pruning and generation for LLM parallel decoding. *arXiv preprint arXiv:2402.13485*.

Inference-cost-aware dynamic tree construction for efficient speculative decoding. (2024). *arXiv preprint*.

Zhou, Y., Zhang, L., and Wang, X. (2024). DySpec: Faster speculative decoding with dynamic token tree structure. *World Wide Web Journal*.

Analysis of draft model size vs. acceptance rate trade-offs in speculative decoding. (2024). *Conference Paper*.

A Technical Appendices and Supplementary Material

Technical appendices with additional results, figures, graphs and proofs may be submitted with the paper submission before the full submission deadline (see above), or as a separate PDF in the ZIP file below before the supplementary material deadline. There is no page limit for the technical appendices.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

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

2. **Limitations**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.

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

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

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: **[TODO]**

   Justification: **[TODO]**

   Guidelines:

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

10. **Broader impacts**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: **[TODO]**

    Justification: **[TODO]**

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

References

[1] 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, 2024. URL `https://arxiv.org/abs/2401.10774`.

[2] Google Research Blog. Looking back at speculative decoding. `https://research.google/blog/looking-back-at-speculative-decoding/`, 2023. Accessed: 2026-01-03.

[3] Yinrong Hong, Zhiquan Tan, and Kai Hu. Inference-cost-aware dynamic tree construction for efficient inference in large language models, 2025. URL `https://arxiv.org/abs/2510.26577`.

[4] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL `https://arxiv.org/abs/2211.17192`.

[5] Medium. Decoding real-time LLM inference: A guide to the latency vs throughput bottleneck. `https://medium.com/learnwithnk/decoding-real-time-llm-inference-a-guide-to-the-latency-vs-throughput-bottleneck-c1ad96442d5` 2024. Accessed: 2026-01-03.

[6] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 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*, ASPLOS '24, pages 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335. URL `https://doi.org/10.1145/3620666.3651335`.

[7] NVIDIA Developer Blog. An introduction to speculative decoding for reducing latency in AI inference. `https://developer.nvidia.com/blog/an-introduction-to-speculative-decoding-for-reducing-latency-in-ai-inference/`, 2023. Accessed: 2026-01-03.

[8] Premai Blog. Transformer inference: Techniques for faster AI models. `https://www.premai.io/blog/transformer-inference-techniques-for-faster-ai-models`, 2024. Accessed: 2026-01-03.

[9] Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025. doi: 10.1162/tacl_a_00735. URL `https://doi.org/10.1162/tacl_a_00735`.

[10] 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, 2024. URL `https://arxiv.org/abs/2401.07851`.

[11] Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, Tianhao Wu, and Lei Zou. Dyspec: Faster speculative decoding with dynamic token tree structure, 2024. URL `https://arxiv.org/abs/2410.11744`.

[12] Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding, 2025. URL `https://arxiv.org/abs/2402.01528`.

[13] Shuzhang Zhong, Zebin Yang, Meng Li, Ruihao Gong, Runsheng Wang, and Ru Huang. Propd: Dynamic token tree pruning and generation for llm parallel decoding, 2024. URL `https://arxiv.org/abs/2402.13485`.