CORRECT WHILE VERIFYING: INTERMEDIATE TARGET PREDICTIONS FOR SPECULATIVE DRAFT CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference, yet its rejection handling remains reactive. Even parallel approaches such as PEARL discover a wrong draft token only after final verification and discard the entire suffix after the first rejection. Target verification is layered: intermediate target computation already yields useful predictions of the correct continuation before final logits are available. We therefore correct drafts while they are being verified. A trainable decoder layer maps intermediate target states to preliminary token distributions whose top-k candidates seed correction branches, which a frozen draft model expands in parallel while the target completes verification. Final target verification authoritatively validates the draft prefix and correction seeds, while branch extensions are queued as candidate drafts for the next cycle, so only target-verified tokens are committed. Under greedy decoding this leaves the target model's verification rule intact. Across Llama, Qwen2.5, and Vicuna pairs our method consistently outperforms PEARL, reaching up to 4.66$\times$ over autoregressive decoding. Code is available at `https://anonymous.4open.science/r/CWV-iclr2027/`.

1 INTRODUCTION

Modern Large Language Models (LLMs) (OpenAI et al., 2024) such as DeepSeek-V3 (DeepSeek-AI, 2024), Qwen3 (Yang et al., 2025), and Llama 3 (Grattafiori et al., 2024) have achieved remarkable success but suffer from high inference latency. Since autoregressive decoding necessitates loading all model parameters for each token, the process has become memory-bound (Leviathan et al., 2023).

Speculative decoding methods (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2024; Zhang et al., 2024a; 2025b; Sadhukhan et al., 2025) address this issue by splitting LLM inference into two phases—draft and verify. Specifically, the *draft* phase generates a sequence of draft tokens at low latency. Subsequently, the *verify* phase utilizes the target model to validate these candidates in a single forward pass. By shifting the bottleneck from memory access to arithmetic computation on the smaller draft model, this paradigm significantly accelerates inference (He et al., 2024; Miao et al., 2024; Du et al., 2024).

Previous speculative decoding efforts have consistently focused on improving the draft. For instance, the EAGLE series (Li et al., 2024a;b) and HASS (Zhang et al., 2025a) leverage features from the target model to enhance draft accuracy, while AdaEAGLE (Zhang et al., 2024b) dynamically predicts the optimal draft length. Similarly, Ouroboros (Zhao et al., 2024) accelerates drafting via cache reuse. Parallel systems such as PEARL (Liu et al., 2025) further overlap drafting and verification to reduce mutual waiting. Yet their rejection handling remains reactive: the system discovers that a token is wrong only after final target verification, and once the first rejection occurs, the entire suffix after that token is discarded. While approaches like Judge Decoding (Bachmann et al., 2025) attempt to relax rejection criteria, they compromise the lossless property of generation.

We propose **Correct-While-Verifying** (**CWV**), a speculative decoding framework that uses target-side intermediate computation to correct drafts before final logits are available. CWV starts from a simple observation: target verification is layered, and intermediate target states often contain useful

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

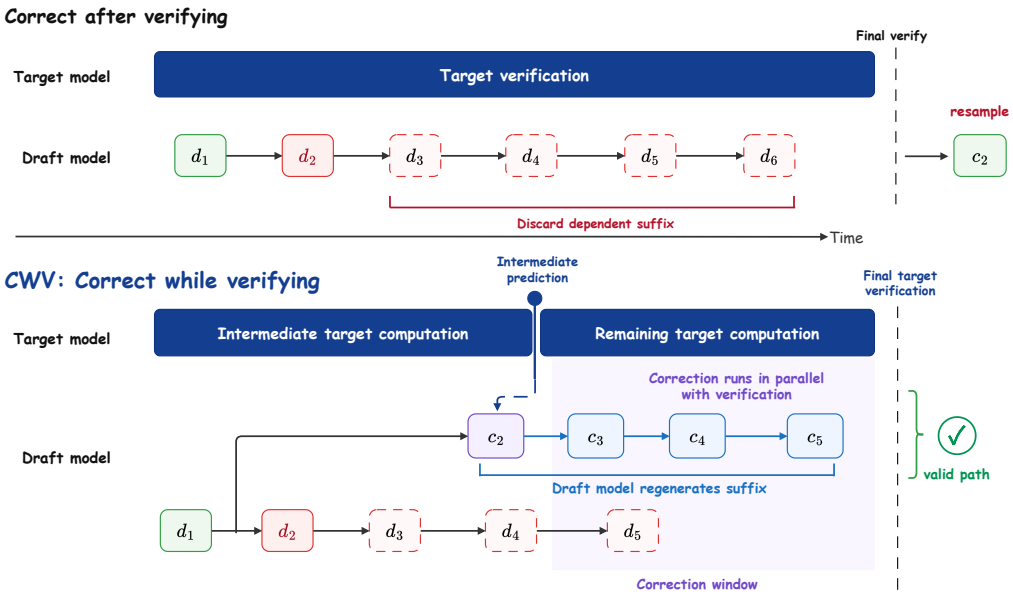

Figure 1: **Correcting while verifying.** In conventional speculative decoding, when a draft token (e.g., d_2) is rejected, all subsequent tokens (d_3, d_4) are discarded. This forces the next step ($T+1$) to restart with only a single corrected token ($\tilde{d}_2$) sampled from the target model. Correct-While-Verifying (CWV) uses intermediate target predictions (L_{d1}) to seed a correction from the erroneous d_2 to c_2. While the target model finishes verifying the current step, a frozen draft model extends this corrected branch with c_3 and c_4. Only the seed c_2 is authoritatively verified in the current step; c_3 and c_4 are queued for Step $T+1$ as candidate drafts, so that the next step starts from a longer verified prefix and needs fewer target calls to reach the same continuation.

predictions about the correct continuation before the final layer finishes. We attach a lightweight trainable decoder layer to these states to produce preliminary token distributions. The top-k candidates from these distributions seed correction branches, and a frozen draft model expands those branches in parallel using a custom attention mask while the target model completes final verification from cached intermediate states. Final target verification then authoritatively validates the original draft prefix together with the correction seeds, while the branch extensions produced in the same step are carried over as candidate drafts for the next cycle; only target-verified tokens are committed. Under greedy decoding this preserves the target model's verification rule.

To summarize, our main contributions are as follows:

- We identify intermediate target computation as a useful source of preliminary predictions for speculative draft correction, reframing rejection handling from a reactive discard step into an online correction opportunity.
- We introduce CWV, which maps intermediate target states to preliminary token distributions, seeds top-k correction branches, and expands them with a frozen draft model under a custom attention mask while final verification continues.
- Across Llama, Qwen2.5, and Vicuna target-draft pairs, CWV consistently improves over PEARL and achieves up to **4.66**$\times$ speedup over autoregressive decoding while preserving the target model's verification rule under greedy decoding.

2 Preliminaries

Notations. We denote the target Large Language Model (LLM) as $\mathcal{M}_t$ and the draft model as $\mathcal{M}_d$. Let $x_{<t}$ represent the prefix sequence generated up to time step t, and $P_{\mathcal{M}}(\cdot|x)$ the distribution over the vocabulary generated by model $\mathcal{M}$ given context x.

Speculative Decoding. Speculative decoding accelerates inference by breaking the dependency on the large target model $\mathcal{M}_t$ for every token, following a *draft-then-verify* paradigm. Given the context $x_{<t}$, the draft model $\mathcal{M}_d$ autoregressively generates γ candidate tokens $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_\gamma\}$, and the target model $\mathcal{M}_t$ processes $[x_{<t}; \tilde{X}]$ in a single parallel forward pass to verify them. Each draft token $\tilde{x}_i$ is verified sequentially using the rejection sampling criterion (or greedy verification), which makes the output distribution match $\mathcal{M}_t$ losslessly for a single proposal chain; we use it in the greedy regime, with acceptance probability

$$\alpha_i = \min\left(1, \frac{P_{\mathcal{M}_t}(\tilde{x}_i | x_{<t}, \tilde{x}_{<i})}{P_{\mathcal{M}_d}(\tilde{x}_i | x_{<t}, \tilde{x}_{<i})}\right). \tag{1}$$

If $\tilde{x}_i$ is accepted it is appended to the output; if rejected, verification terminates at i and $\mathcal{M}_t$ resamples a corrected token $\hat{x}_i$ from the adjusted distribution.

Parallel Speculative Decoding. While sequential methods operate in a strict serial order, parallel approaches such as PEARL (Liu et al., 2025) significantly improve throughput by decoupling drafting and verification (Monea et al., 2023; Xiao et al., 2024; Shen et al., 2026; An et al., 2026; Lin et al., 2025). Specifically, PEARL splits verification into a *Pre-verify* phase (validating the first token from the previous step) and a *Post-verify* phase (validating the subsequent draft sequence). This schedule masks part of the draft model latency behind target-model computation. Despite this parallelism, PEARL still treats rejection as a final-verification event: it only knows that a token is wrong after the target model produces final logits, and it still discards the suffix after the first rejected token.

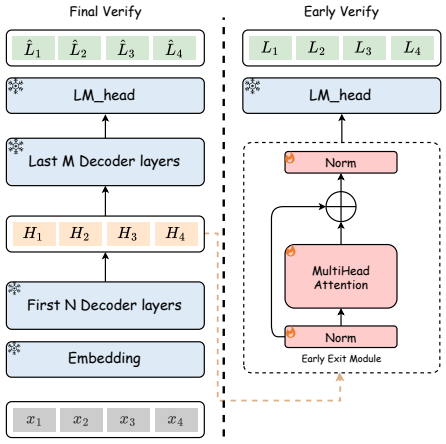

Figure 2: Architecture of CWV. A lightweight trainable decoder layer maps intermediate target hidden states from the first N layers to preliminary token distributions, while the cached hidden states are reused by final target verification. "Snow" denotes frozen parameters and "fire" denotes trainable parameters.

The Bottleneck of Reactive Rejection Handling. Despite success of speculative decoding, its draft-then-verify paradigm suffers from a structural inefficiency caused by reactive rejection handling. Let $r \in \{1, \ldots, \gamma\}$ be the index of the first rejected token (i.e., the first index where sampling fails). While the system successfully accepts the prefix $\tilde{x}_{<r}$, the strict sequential dependency forces the immediate discard of the entire suffix. We formally define the Wasted Set $\mathcal{W}$ as:

$$\mathcal{W} = \{\tilde{x}_{r+1}, \ldots, \tilde{x}_\gamma\} \tag{2}$$

Although the draft model expended compute to generate $\mathcal{W}$, and the target model expended memory bandwidth to verify $\mathcal{W}$, these tokens contribute zero effective length. Critically, during the target model's verification of this doomed suffix, the draft model remains idle or suffers from invalid inference, while the computational resources consumed to generate and verify this discarded suffix are wasted. This limitation motivates CWV, which uses intermediate target predictions to correct likely rejection points before final verification completes.

3 Method

3.1 Motivation

The formulation in Section 2 highlights a critical inefficiency: the *computation gap* (Eq. 2) where the draft model remains idle while the target model verifies a doomed suffix $\mathcal{W}$. Current parallel methods such as PEARL (Liu et al., 2025) attempt to hide latency through pipelining, yet they remain fundamentally reactive because they can only detect errors after the full forward pass. Our

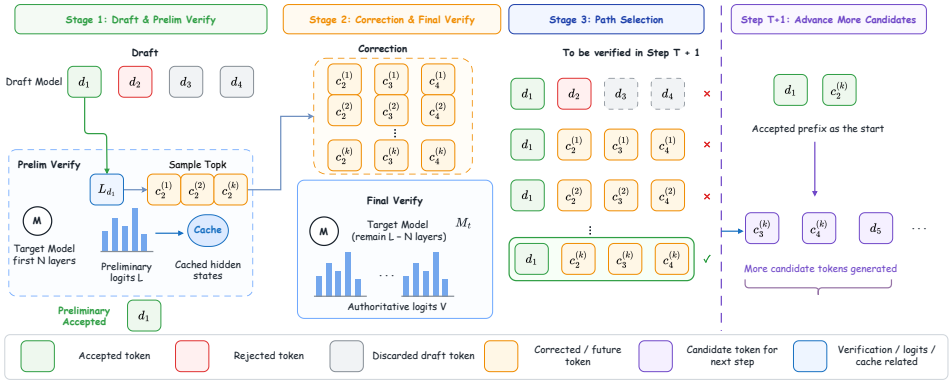

Figure 3: One CWV cycle. **Stage 1** drafts $\{d_1, \ldots, d_4\}$ with the frozen draft model while the target model's first N layers produce the preliminary logits L_{early} and cache the intermediate hidden states; d_2 is erroneous. **Stage 2** expands one branch $\{d_1, c_2^{(j)}, c_3^{(j)}, c_4^{(j)}\}$ per top-k candidate $c_2^{(j)}$ under a custom attention mask, concurrently with the target model's remaining layers producing the authoritative logits. **Stage 3** selects the branch whose seed agrees with the authoritative logits. Only the verified prefix d_1 and the accepted seed $c_2^{(j)}$ are committed; the extensions $c_3^{(j)}, c_4^{(j)}$ are queued as candidate drafts for Step $T+1$. Color legend: green = current prefix, blue = original draft, yellow = correction seeds, orange = branch extensions, grey = discarded draft tokens.

motivation stems from the insight that target verification is not atomic but layered: intermediate target states often contain enough information to predict plausible continuations before final logits are available, so the idle time identified in Eq. 2 is not merely a penalty but a window for correction. This leads to the core design philosophy of CWV: **correct drafts while target verification is still running.** Instead of passively waiting for the final rejection, CWV (i) probes intermediate target states with a trainable decoder layer to obtain preliminary token distributions, and (ii) uses the top-k preliminary candidates to seed correction branches that a frozen draft model expands before final verification concludes. By transforming dead time after rejection into useful branch expansion, CWV converts potential discarded suffixes into candidate continuations that can be validated by the target model.

3.2 The CWV Framework

Based on the motivation above, CWV augments parallel speculative decoding with intermediate target prediction and correction branch expansion. We assume an inference setup with a target model $\mathcal{M}_t$ equipped with a trainable decoder layer inserted after the first N layers and a lightweight frozen draft model $\mathcal{M}_d$. The workflow processes an initial sequence $x_{<t}$ as follows:

Parallel Drafting and Intermediate Prediction. Instead of waiting for the full target model, we decouple the process into two parallel streams:

- **Draft:** The draft model $\mathcal{M}_d$ autoregressively generates a short draft sequence $\tilde{X}$ of length γ_1.
- **Intermediate target prediction:** Simultaneously, the target model $\mathcal{M}_t$ processes $x_{<t}$ through its first N layers. The trainable decoder layer maps the resulting states to preliminary token distributions L_{early}, and the intermediate hidden states H are cached for the final target pass.

Upon synchronization, CWV has both the draft sequence and target-side preliminary predictions about the correct continuation.

Parallel Correction and Final Verification. Leveraging the preliminary distributions L_{early}, CWV performs immediate draft correction without discarding the current step:

- **Correction branch expansion:** We construct a candidate set $\mathcal{C}$ by selecting the top-k tokens from L_{early}. We then form a branch-structured input x_{branch} combining the prefix $x_{<t}$, the original draft $\tilde{X}$, and the correction seeds $\mathcal{C}$. Under a **custom attention mask**, the frozen draft model generates subsequent tokens $\tilde{X}_{\text{corr}}$ of length γ_2 for multiple branches in parallel. Figure 3 illustrates one step of this pipeline. The draft model produces the draft $\tilde{X} = \{d_1, \ldots, d_4\}$, in which d_2 is an erroneous token that the final target verification will later reject. From the intermediate states, the trainable decoder layer emits L_{early} and its top-k candidates $c_2^{(1)}, \ldots, c_2^{(k)}$ replace d_2 as correction seeds. Each seed opens a separate branch $\{d_1, c_2^{(j)}, c_3^{(j)}, c_4^{(j)}\}$, which the frozen draft model extends by γ_2 further tokens (**Stage 2**). The mask gives every branch its own visible context: branch j attends to the shared prefix and to the preceding tokens of the *same* branch, never to another branch, so its KV entries live in a private partition of the cache and are not shared with $j' \neq j$. All branches therefore expand in a single batched forward pass, and the expansion never waits for the target model's remaining layers because the draft-side computation depends only on the frozen draft model. This ensures that even if $\tilde{X}$ is flawed, a correct alternative path is likely being generated simultaneously.
- **Final target verification:** Concurrently, the target model resumes computation from the cached states H, propagating through the remaining layers to produce the authoritative logits L_{target}.

Path Selection and Commit Boundary. We make precise which tokens CWV commits in a cycle and which it merely queues. **Authoritatively verified in this step.** The target logits L_{target} cover only the original draft path and the seeds: in Stage 3 of Figure 3 the row $\{d_1, d_2, \ldots\}$ is rejected because the target distribution disagrees with d_2, and a row $\{d_1, c_2^{(j)}, c_3^{(j)}, c_4^{(j)}\}$ is selected when the target distribution at that position agrees with the seed $c_2^{(j)}$. The verified consensus is the prefix $\{d_1\}$ plus the accepted seed $c_2^{(j)}$. **Queued for $T+1$.** The extensions $c_3^{(j)}, c_4^{(j)}$ are *not* verified here: target verification covers the draft and the seeds. Appending a winning branch to the prefix puts them into the speculative tail, and the single target forward of the next cycle verifies that tail together with the seed, as the "To be verified in Step $T+1$" label of Figure 3 indicates. **Committed in this cycle.** Only the verified part is appended to the output, so the extensions only seed a longer draft for the next cycle: CWV reduces the number of steps needed to reach a given continuation rather than committing more tokens per step. Since no unverified token is ever committed, the output is always a subsequence of what the target model would produce on its own.

The complete inference procedure is summarized in Algorithm 1 in Appendix D.

3.3 Theoretical Analysis

To quantify the efficiency gains of CWV, we analyze the system throughput and derive a cost model comparing it to existing parallel decoding baselines.

3.3.1 Assumptions

We assume a greedy decoding setting ($temperature = 0$), under which the target sequence is deterministic for a given input and the positions where rejections occur are fixed for a fixed draft model. For standard parallel speculative decoding this makes the total number of verification steps, denoted C_{total}, constant; CWV sets out to break that limitation by correcting erroneous draft tokens on the fly.

3.3.2 Computational Cost Analysis

We adapt the cost analysis framework from PEARL to model the verification overhead, where t is the current verification step and n_{t-1} the number of tokens accepted in step $t-1$.

Baseline Verification Strategy. Let γ denote the standard draft length of the baseline schedule. In standard parallel decoding (e.g., PEARL), the strategy for step t depends on the success of step $t-1$ and switches between two modes: **Post-verify**, which verifies a full draft of length γ when

$n_{t-1} = \gamma$; and **Pre-verify**, which verifies only the immediate next token (effectively length 1) to recover the context when $n_{t-1} < \gamma$. The cost is therefore dominated by the counts of these operations, $C_{\text{total}} = C_{\text{post}} + C_{\text{pre}}$.

CWV Verification Strategy. CWV intervenes in the *Pre-verify* scenario. When $n_{t-1} < \gamma$, instead of resigning to a low-efficiency Pre-verify step, it activates intermediate target prediction and correction branch expansion: if the trainable decoder layer predicts the correct token among its top-k candidates, CWV generates and verifies a corrected sequence, and otherwise it falls back to the standard Pre-verify behaviour. Let C_{early} be the number of successful early corrections and $\hat{C}_{\text{pre}}$ the residual failed Pre-verify operations, so that $C_{\text{pre}} = C_{\text{early}} + \hat{C}_{\text{pre}}$ (Eq. 3) and the number of verification calls is $C_{\text{total}} = C_{\text{post}} + C_{\text{early}} + \hat{C}_{\text{pre}}$ (Eq. 4).

$$C_{\text{pre}} = C_{\text{early}} + \hat{C}_{\text{pre}} \tag{3}$$

$$C_{\text{total}} = C_{\text{post}} + C_{\text{early}} + \hat{C}_{\text{pre}} \tag{4}$$

Efficiency Derivation. The total number of verified tokens N_{verified} is the accumulated length from all steps. In CWV, a successful C_{early} step contributes γ_2 verified tokens (the verified prefix and the accepted correction seed, together with the queued extensions that the next step re-verifies), whereas a standard $\hat{C}_{\text{pre}}$ step contributes only 1.

$$N_{\text{verified}} = \gamma \cdot C_{\text{post}} + \gamma_2 \cdot C_{\text{early}} + 1 \cdot \hat{C}_{\text{pre}} \tag{5}$$

Substituting Eq. 4 and Eq. 3 into Eq. 5 and rearranging gives the exact identity

$$C_{\text{total}} = \frac{N_{\text{verified}}}{\gamma} \; - \; (\gamma_2 - 1)\, C_{\text{early}} \; + \; \frac{\gamma - 1}{\gamma}\big(N_{\text{verified}} - \gamma\, C_{\text{post}}\big). \tag{6}$$

The merit of Eq. 6 is visible when we compare CWV against the un-corrected schedule *at equal generative work*: holding N_{verified} and C_{post} fixed (that is, the same number of tokens produced and the same number of full-length Post-verify steps), setting $C_{\text{early}} = 0$ recovers the baseline, and therefore

$$C_{\text{total}}^{\text{baseline}} - C_{\text{total}}^{\text{CWV}} = (\gamma_2 - 1)\, C_{\text{early}}. \tag{7}$$

Every successful early correction therefore removes $(\gamma_2 - 1)$ verification calls: it converts γ_2 low-yield Pre-verify steps of length 1 into a single call, and since $\gamma_2 > 1$ the saving is strictly positive.

We discuss the limits of this model, in particular that it counts verification calls rather than arithmetic work, in Appendix D.2.

4 Experiments

Unlike standalone speculative decoding architectures, CWV is a plug-in correction mechanism for parallel speculative decoding. Its acceleration capability is therefore linked to the base scheduler and the specific draft–target model pair. In this section, we evaluate CWV by integrating it with PEARL (Liu et al., 2025). Since CWV trains only a lightweight decoder layer and freezes both the draft and target models, we primarily compare it against training-free baselines. We also include EAGLE2 and EAGLE3 results as training-based references.

Models. We evaluate three Llama target–draft pairs: Llama2-70B with Llama2-7B (Touvron et al., 2023), Llama3.1-70B with Llama3.1-8B, and Llama3.1-70B with Llama3.2-1B (Grattafiori et al., 2024; Meta AI, 2024). Both models in each pair remain frozen, and only the intermediate decoder layer introduced by CWV is trained. Appendix D.5 discusses the choice of model pairs and evaluates scalability on a smaller target model.

Datasets. We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024) (the 500-problem held-out subset of MATH (Hendrycks et al., 2021)), AIME 2024 (Mathematical Association of America, 2024), and AIME 2025 (Mathematical Association of America, 2025), and code generation on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). MT-bench (Zheng et al., 2023) assesses multi-round dialogue categories.

Table 1: Experiment results on GSM8K, HumanEval, AIME 2024, AIME 2025, MATH500, and MBPP. Speedup is computed as $\tau = S_{\text{method}}/S_{\text{AR}}$ using the autoregressive speed for the same target model. We bold the best results for each model–benchmark combination. Assisted Generation and Speculative Decoding are reproduced with their official implementations. EAGLE2 and EAGLE3 use their official released weights, evaluated by us on an H800 on their native Llama3.3-70B target model rather than quoted from their papers, and are included as out-of-target-model references only. Methods without available results on a specific dataset are marked with "–".

		GSM8K		HumanEval		AIME 2024		AIME 2025		MATH500		MBPP	
Model	**Method**	speed(token/s)	τ	speed(token/s)	τ	speed(token/s)	τ	speed(token/s)	τ	speed(token/s)	τ	speed(token/s)	τ
Llama2 7B-70B	Autoregression	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×
	Speculative Decoding	29.46	2.14×	30.35	2.20×	26.79	1.95×	28.18	2.05×	28.78	2.09×	28.53	2.07×
	Assisted Generation	23.31	1.69×	26.15	1.90×	20.83	1.51×	18.29	1.33×	19.08	1.39×	20.42	1.48×
	PEARL	40.91	2.97×	38.82	2.82×	33.30	2.42×	32.50	2.36×	35.85	2.60×	35.89	2.61×
	CWV	**41.30**	**3.00**×	**41.04**	**2.98**×	**34.64**	**2.52**×	**34.18**	**2.48**×	**36.11**	**2.62**×	**37.44**	**2.72**×
Llama3.1-8B Llama3.1-70B	Autoregression	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×
	Speculative Decoding	29.99	2.02×	33.14	2.23×	27.18	1.83×	26.93	1.81×	28.79	1.94×	31.83	2.14×
	Assisted Generation	34.42	2.32×	35.90	2.42×	28.94	1.95×	29.68	2.00×	31.64	2.13×	29.87	2.01×
	PEARL	45.79	3.08×	45.81	3.08×	37.74	2.54×	41.26	2.78×	44.61	3.00×	41.91	2.82×
	CWV	**46.00**	**3.10**×	**47.56**	**3.20**×	**38.94**	**2.62**×	**41.61**	**2.80**×	**44.92**	**3.02**×	**43.44**	**2.93**×
Llama3.2-1B Llama3.1-70B	Autoregression	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×
	Speculative Decoding	39.79	2.68×	48.03	3.23×	34.09	2.30×	33.13	2.23×	38.14	2.57×	38.70	2.61×
	Assisted Generation	47.03	3.17×	50.73	3.42×	38.98	2.62×	40.34	2.72×	45.61	3.07×	38.10	2.57×
	PEARL	52.83	3.56×	61.10	4.11×	42.19	2.84×	43.81	2.95×	52.49	3.53×	41.03	2.76×
	CWV	**59.41**	**4.00**×	**67.30**	**4.53**×	**48.07**	**3.24**×	**50.45**	**3.40**×	**58.22**	**3.92**×	**50.89**	**3.43**×
Llama3.3-70B EAGLE2		36.53	2.46×	46.48	3.13×	34.40	2.32×	34.00	2.29×	35.90	2.42×	35.90	2.42×
Llama3.3-70B EAGLE3		63.30	4.39×	65.80	4.38×	50.90	3.35×	49.90	3.34×	54.90	3.59×	63.90	4.25×

How often the correction fires. We report the correction trigger rate, the share of verification rounds carried by a correction branch rather than the original draft, in Appendix D.3 (Table 7). It rises from about one round in eight for the well-matched Llama3.1-8B draft to roughly a third for the weakest.

Baselines. We use vanilla **Autoregressive Decoding** as the $1.00\times$ reference, and reproduce **Speculative Decoding (SD)** (Leviathan et al., 2023) and **Assisted Generation** (Gante, 2023) with their official implementations. Since CWV builds on PEARL's parallel schedule (Liu et al., 2025), PEARL is our primary training-free baseline. We additionally evaluate **EAGLE2** (Li et al., 2024b) and **EAGLE3** (Li et al., 2025) using their official released weights on their native Llama3.3-70B target model, since published speedups of learning-based drafters depend strongly on corpus and hardware.

Metrics. Our primary metric is the *Speedup Ratio* $\tau = S_{\text{method}}/S_{\text{AR}}$ (tokens/s relative to vanilla autoregressive decoding of the same target model) together with the average acceptance length (Section 4.1.1). Since CWV uses the target model's own verification rule, generation quality is preserved by construction under greedy decoding, so we focus on latency and acceptance. Autoregressive throughput is a property of the target model and hardware rather than of the dataset, so S_{AR} is measured once per target model.

Implementation Details. All experiments run under greedy decoding ($\text{temperature} = 0$) with `bfloat16` weights and KV cache and batch size 1, using three NVIDIA H800 (80GB) GPUs for the 70B target model and one further H800 for the frozen draft model. Drafting, correction and target verification are dispatched on separate CUDA streams so that branch expansion overlaps the remaining target layers. Throughput is measured per sample as generated tokens over wall-clock time and aggregated as $\sum$tokens/$\sum$time over the full split; because per-sample throughput varies substantially with output length and prompt, the aggregate carries a standard error of 2–4% depending on split size (per-sample coefficient of variation 17–26%). The trainable decoder layer is inserted after layer $N = L/2$ (L = number of target layers) and is trained on ShareGPT (anon8231489123, 2023) with AdamW at a learning rate of 10^{-3} with cosine decay and weight decay 0.1 for 3 epochs, following Medusa's training protocol (Cai et al., 2024). Unless stated otherwise we use $k = 4$ candidates.

4.1 Main Results

Table 1 reports results on the six reasoning and code-generation benchmarks. CWV outperforms Speculative Decoding and Assisted Generation across all three Llama model pairs, and improves on

Table 2: Speed (token/s) and speedup ($\tau = S_{\text{method}}/S_{\text{AR}}$) results on MT-bench. We bold the best speed for each model combination. The baseline Autoregression (AR) speed is 13.77 token/s for Llama2-70B and 14.85 token/s for Llama3.1-70B. **S**: Speed, τ: Speedup ratio.

		Writing		**Roleplay**		**Reasoning**		**Math**		**Coding**		**Extraction**		**STEM**		**Humanities**		**Average**	
Pair	**Method**	token/s	τ	token/s	τ	token/s	τ	token/s	τ	token/s	τ	token/s	τ	token/s	τ	token/s	τ	token/s	τ
Llama2 7B-70B	AR	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×	13.77	1.00×
	SD	25.95	1.88×	24.74	1.80×	28.98	2.10×	33.83	2.46×	35.18	2.55×	34.84	2.53×	26.73	1.94×	25.95	1.88×	29.51	2.14×
	PEARL	30.08	2.18×	32.45	2.36×	33.39	2.42×	37.97	2.76×	35.80	2.60×	36.69	2.66×	34.45	2.50×	34.24	2.49×	34.38	2.50×
	CWV	**34.38**	**2.50**×	**34.85**	**2.53**×	**36.99**	**2.69**×	**41.17**	**2.99**×	**38.76**	**2.81**×	**40.58**	**2.95**×	**37.83**	**2.75**×	**36.59**	**2.66**×	**37.67**	**2.74**×
Llama3.1-8B Llama3.1-70B	AR	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×
	SD	25.48	1.72×	24.32	1.64×	24.69	1.66×	28.95	1.95×	30.37	2.05×	27.62	1.86×	26.96	1.82×	25.73	1.73×	26.72	1.80×
	PEARL	32.66	2.20×	32.32	2.18×	33.97	2.29×	43.88	2.95×	44.24	2.98×	41.79	2.81×	35.92	2.42×	35.72	2.41×	37.56	2.53×
	CWV	**36.20**	**2.44**×	**34.99**	**2.36**×	**37.09**	**2.50**×	**44.93**	**3.03**×	**45.74**	**3.08**×	**43.36**	**2.92**×	**38.43**	**2.59**×	**37.26**	**2.51**×	**39.77**	**2.68**×
Llama3.2-1B Llama3.1-70B	AR	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×	14.85	1.00×
	SD	28.92	1.95×	26.65	1.79×	32.01	2.16×	41.47	2.79×	44.64	3.01×	40.99	2.76×	28.67	1.93×	27.56	1.86×	33.83	2.28×
	PEARL	32.57	2.19×	27.68	1.86×	34.75	2.34×	55.28	3.72×	56.60	3.81×	48.94	3.30×	31.27	2.11×	29.59	1.99×	39.63	2.67×
	CWV	**38.98**	**2.62**×	**36.28**	**2.44**×	**37.77**	**2.54**×	**66.79**	**4.50**×	**69.19**	**4.66**×	**57.81**	**3.89**×	**43.62**	**2.94**×	**38.93**	**2.62**×	**48.73**	**3.28**×

Table 3: Experiment results on Qwen2.5-7B/72B. Speed is measured in tokens/s.

Method	**GSM8K**		**HumanEval**	
	Speed	τ	Speed	τ
Autoregression	14.01	1.00×	14.01	1.00×
PEARL	32.27	2.30×	38.86	2.77×
CWV	**34.02**	**2.43**×	**39.87**	**2.85**×

PEARL in all eighteen model–benchmark combinations. The gains grow as the drafter gets weaker, which is consistent with the mechanism: on the Llama3.2-1B/Llama3.1-70B pair CWV reaches 4.53× on HumanEval against PEARL's 4.11×, whereas the CWV–PEARL margin is smallest for the well-matched Llama3.1-8B draft, where few drafts are rejected in the first place.

Table 2 shows the MT-bench results for the same model pairs. CWV is faster than PEARL in every dialogue category and model pair, with the largest margin on the coding and math categories, indicating that intermediate target prediction and correction-branch expansion improve throughput without introducing prohibitive synchronization overhead.

Furthermore, to validate CWV's generalizability beyond the Llama family, we use Qwen2.5-7B as the draft model and Qwen2.5-72B as the target model (Qwen Team, 2024). Table 3 summarizes the results.

As shown, CWV consistently outperforms PEARL on the Qwen2.5 architecture, achieving speedups of 2.43× on GSM8K and 2.85× on HumanEval. This confirms that correction from intermediate target predictions generalizes beyond Llama-based models. It is worth noting that Qwen2.5 possesses an exceptionally large vocabulary due to its merged token strategy, which inherently limits the upper bound of acceleration achievable through speculative decoding. Therefore, the speedup ratios are within the expected range for this architecture, and CWV still delivers consistent improvements over the baseline.

For the learned drafters, we do not quote numbers from their papers. EAGLE-2 and EAGLE-3 are trained against a different target model than ours (Llama3.3-70B-Instruct rather than Llama3.1-70B), are evaluated on different benchmark subsets, and their absolute speedups depend on the training corpus and hardware, so numbers taken from their reports would not be comparable to those we obtain on our own hardware. We therefore evaluate both methods with their official released weights on their native target model, using the same H800 setup, greedy decoding and evaluation harness as every other row in Table 1; the reported EAGLE-2 and EAGLE-3 speedups are consequently measured by us rather than published figures, and they remain an out-of-target-model reference rather than a controlled comparison. Under this protocol CWV attains acceleration comparable to or exceeding EAGLE-3 on several of the mathematical benchmarks while training only a single lightweight decoder layer on ShareGPT rather than a dedicated drafter.

We additionally evaluate Vicuna target-draft pairs in Appendix D.5.

4.1.1 AVERAGE ACCEPTANCE LENGTH

Table 4 reports the average number of tokens advanced between two consecutive rejections, where a rejection means either a draft token or an early-exit token being turned down. This run length, rather than the number of steps on its own, governs how many steps a given continuation requires. CWV advances 1.9–3.4× more tokens between rejections than PEARL: a Pre-verify step that advances a single token becomes a step that advances the corrected branch, and the following steps then have only a short suffix left to verify instead of a full draft. This does *not* transfer one-to-one to throughput. The target side costs no more: a correction keeps the single forward per cycle and simply narrows the window it verifies. The extra cost is on the draft side, where expanding the k branches multiplies drafting; dividing the commit ratios by the throughput ratios of Table 1 implies a CWV step costs about 2.7–3.1× more on GSM8K and HumanEval but only 1.5× more on MBPP. CWV thus trades fewer steps with more draft-side work for PEARL's more numerous cheap ones, and it is the resulting speedup, not the commit ratio alone, that Table 1 reports.

Table 4: Average number of tokens committed per verification step on the Llama3.2-1B&Llama3.1-70B model pair. Only tokens that the target model has authoritatively verified are counted.

Method	GSM8K	HumanEval	MBPP
PEARL	9.53	15.44	12.19
CWV	**32.73**	**45.25**	**23.22**

4.1.2 TRAINABLE DECODER LAYER ACCURACY

To provide a direct assessment of the trainable decoder layer's predictive quality, we evaluated its accuracy across top-1 to top-5 settings. Using the target model's full forward pass outputs as ground truth, we measured how often the correct token (the one ultimately accepted by the target model) falls within the top-k candidates from the preliminary distributions. The results are shown in Table 5.

Table 5: Top-k accuracy of the trainable decoder layer on GSM8K and HumanEval.

Dataset	Top-1	Top-2	Top-3	Top-4	Top-5
GSM8K	0.64	0.75	0.80	0.82	0.84
HumanEval	0.61	0.73	0.78	0.82	0.83

As shown, while the Top-1 accuracy is moderate (0.61–0.64), the accuracy rapidly improves as k increases. Under the top-4 setting used in our main experiments, the accuracy reaches 0.82, meaning that in 82% of cases, the trainable decoder layer identifies the target token among its top-4 candidates. This high recall justifies CWV's top-k strategy: by generating multiple correction branches from these candidates, the draft model is likely to produce a valid continuation even when the original draft token is erroneous.

5 RELATED WORK

CWV sits between learning-based and self-speculative drafters that draft with the target model itself (Kangaroo, LayerSkip, Medusa, the EAGLE series, HASS), early-exit methods that terminate inference at an intermediate layer, and PEARL, whose parallel schedule it extends. What distinguishes it is where the auxiliary computation goes: those methods spend it on *proposing* tokens earlier in the same schedule, whereas CWV trains one decoder layer and spends it on repairing a draft the scheduler has already rejected. Appendix D.1 gives the detailed discussion and a positioning table.

6 CONCLUSION AND FUTURE WORK

We presented Correct-While-Verifying, which uses intermediate target computation to correct drafts while final verification runs. Across Llama, Qwen2.5 and Vicuna pairs it improves on PEARL consistently, reaching up to 4.66× under greedy decoding. Future work will improve the quality-latency trade-off of the intermediate predictions.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM inference with low-cost PARallel draft model adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 144562–144579, 2026.

anon8231489123. ShareGPT_Vicuna_unfiltered. https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered, 2023. Community-shared mirror of the ShareGPT conversations; used as the training set for the auxiliary decoder layer.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtSSFiqW6y.

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 5910–5924. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.362. URL https://doi.org/10.18653/v1/2023.emnlp-main.362.

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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

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

DeepSeek-AI. Deepseek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Cunxiao Du, Jing Jiang, Yuanchen Xu, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. GliDe with a CaPE: A low-hassle method to accelerate speculative decoding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11704–11720. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24c.html.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.681. URL http://dx.doi.org/10.18653/v1/2024.acl-long.681.

Joao Gante. Assisted generation: a new direction toward low-latency text generation. https://huggingface.co/blog/assisted-generation, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.88. URL https://aclanthology.org/2024.naacl-long.88/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=4exx1hUffq.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. BiTA: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305, 2025. doi: 10.1016/j.eswa.2025.127305.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11946–11965. Curran Associates, Inc., 2024a. doi: 10.52202/079017-0381.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster LLM inference with thompson sampling control mechanism. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3027–3043. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.findings-acl.179.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. PEARL: Parallel speculative decoding with adaptive draft length. In *The Thirteenth International Conference on Learning Representations*, volume 2025, pp. 1085–1104, 2025. URL https://openreview.net/forum?id=QOXrVMiHGK.

Mathematical Association of America. AIME 2024. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024, 2024.

Mathematical Association of America. AIME 2025. https://huggingface.co/datasets/yentinglin/aime_2025, 2025.

Meta AI. Llama 3.2: Revolutionizing edge AI with open, customizable multimodal models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, 2024. Introduces the Llama 3.2 1B/3B text models and the 11B/90B vision models.

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*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335. URL https://doi.org/10.1145/3620666.3651335.

Giovanni Monea, Armand Joulin, and Edouard Grave. PaSS: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023. Oral at the Third Workshop on Efficient Natural Language and Speech Processing (ENLSP-III), NeurIPS 2023.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Qwen Team. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 6835–6850, 2025.

Yuhao Shen, Junyi Shen, Quan Kong, Tianyu Liu, Yao Lu, and Cong Wang. SpecBranch: Speculative decoding via hybrid drafting and rollback-aware branch parallelism. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BrnlCSqO6n.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31, pp. 10086–10095. Curran Associates, Inc., 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi,

Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Yeshwanth Venkatesha, Souvik Kundu, and Priyadarshini Panda. Fast and cost-effective speculative edge-cloud decoding with early exits. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=PTIUjARnbc.

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. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.456.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. ParallelSpec: Parallel drafter for efficient speculative decoding. *arXiv preprint arXiv:2410.05589*, 2024.

Canwen Xu and Julian McAuley. A survey on dynamic neural networks for natural language processing. In *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2370–2381, Dubrovnik, Croatia, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.180. URL https://aclanthology.org/2023.findings-eacl.180.

Jiaming Xu, Jiayi Pan, Yongkang Zhou, Siming Chen, Jinhao Li, Yaoxiu Lian, Junyi Wu, and Guohao Dai. SpecEE: Accelerating large language model inference with speculative early exiting. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pp. 467–481, 2025. doi: 10.1145/3695053.3730996.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Chen Zhang, Zhuorui Liu, and Dawei Song. Beyond the speculative game: A survey of speculative execution in large language models. *arXiv preprint arXiv:2404.14897*, 2024a.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*, volume 2025, pp. 35367–35388, 2025a. URL https://openreview.net/forum?id=1bEEpmz2tV.

Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S Yu, et al. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*, 2025b.

Situo Zhang, Hankun Wang, Da Ma, Zichen Zhu, Lu Chen, Kunyao Lan, and Kai Yu. Adaeagle: Optimizing speculative decoding via explicit modeling of adaptive draft structures, 2024b. URL https://arxiv.org/abs/2412.18910.

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13378–13393. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.742.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. URL https://openreview.net/forum?id=uccHPGDlao. Datasets and Benchmarks Track.

A Ethics Statement

We properly reference all prior methods and datasets employed in our study, using exclusively publicly available data without any utilization of private information. Moreover, we carefully maintain our developed inference acceleration techniques, ensuring their implementation remains free from any discriminatory effects.

B AI Use Statement

In this work, we used generative AI tools (GPT-5.6-sol and DeepSeek V4 Pro) for editing the manuscript to improve readability and grammar. We have not used generative AI tools for generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims or proofs, proposing or refining hypotheses, designing or providing feedback on our research methodology or experiments, implementing methods, cleaning or reformatting datasets, or interpreting results, and the remaining tasks with required disclosure are not applicable to this work. We have reviewed all AI-assisted work: every edit was checked by the authors against the experiments and results it describes. We take responsibility for the final content of this work, including text, claims and artifacts.

C Reproducibility Statement

We provide the complete implementation—the trainable decoder layer, the custom attention mask used for parallel branch expansion, and the evaluation harness—in an anonymous repository at `https://anonymous.4open.science/r/CWV-iclr2027/`. All datasets used in our experiments (GSM8K, MATH500, AIME 2024, AIME 2025, MBPP, HumanEval and MT-bench) and the ShareGPT corpus used to train the decoder layer are publicly available. Training and inference settings, including hardware, precision, batch size, the insertion depth of the decoder layer, the optimizer schedule and the number of correction candidates, are listed in the Implementation Details paragraph of Section 4. The branching strategy is specified in Section 3.2 and formalised in Algorithm 1; the assumptions and limits of the cost model are discussed in Appendix D.2, the memory overhead of the custom attention mask is quantified in Appendix D.4, and correction trigger rates together with the additional results on 13B targets are reported in Appendices D.3 and D.5.

D Additional Clarifications and Experiments

In this appendix, we address specific implementation details regarding the branching strategy, provide a quantitative analysis of the memory overhead introduced by the custom attention mask, and present additional evaluation results on long-context datasets.

D.1 Extended Related Work

Early-exit techniques. Early-exit techniques stop inference at an intermediate layer for simpler inputs, balancing the capacity needed for complex tasks against the efficiency of shallower networks on "easy" examples (Xu & McAuley, 2023; Xu et al., 2025; Venkatesha et al., 2025). Bae et al. (2023) proposes a fast and robust early-exiting framework with synchronized parallel decoding, and Liu et al. (2024b) adaptively controls the number of drafted tokens via Thompson sampling.

Learning-based and self-speculative drafters. Closest to CWV are methods that draft with the target model itself. Kangaroo (Liu et al., 2024a) reuses a shallow sub-network of the target as a self-draft model with a lightweight adapter, and LayerSkip (Elhoushi et al., 2024) trains early-exit layers and reuses the shallower exit for self-speculative decoding, so that later target layers verify the exit's proposals. Medusa (Cai et al., 2024) attaches lightweight prediction heads and validates their candidates under a tree-structured mask. The EAGLE series (Li et al., 2024a;b; 2025) instead trains a drafter on target features, and HASS (Zhang et al., 2025a) aligns drafter and target representations. These methods differ from CWV in what is trained and where the auxiliary computation is spent: they train an exit layer, heads or a drafter and spend that computation on *proposing* tokens earlier

within the same schedule, whereas CWV trains a single decoder layer, keeps the frozen draft model, and spends the computation on repairing a draft the scheduler has already rejected. CWV is thus complementary to these drafters rather than a competitor, at the cost of keeping the draft model resident alongside the target; comparing against learned drafters requires controlling for target model and training corpus, as we do in Section 4. Table 6 summarises the positioning.

Table 6: Positioning of CWV against the closest lines of work.

Method	What is trained	Extra model resident?	Verification rule	Where the extra compute goes
LayerSkip	early-exit layers	no (target weights)	self-speculative, then full verify	proposing earlier
Medusa	multiple prediction heads	heads only	tree-structured parallel verify	proposing earlier
EAGLE-2/3	autoregressive drafter	yes (drafter)	standard speculative verify	proposing earlier
Kangaroo	shallow exit + adapter	no (target weights)	self-speculative, then full verify	proposing earlier
CWV (ours)	one decoder layer	yes (frozen draft)	target verification, greedy-lossless	repairing an already-rejected draft

PEARL. PEARL (Liu et al., 2025) introduces a pre-verification strategy to validate the initial draft token during the drafting phase and a post-verification approach to generate additional draft tokens during the verification phase. By implementing these strategies, PEARL enables parallel execution of the drafting and verification phases, while adaptively adjusting the draft length to suit different scenarios. CWV is complementary to this scheduling idea: it keeps PEARL's overlap, but uses intermediate target predictions to create correction branches before final verification reveals the first rejection.

D.2 Limits of the Cost Model

Three limits matter for how Eq. 7 should be read. First, we count *verification calls*, not arithmetic work: the early-layer projection, the k-fold branch expansion and the wasted work of failed corrections are assumed negligible against one target forward pass, so we present Eq. 7 as an explanatory model rather than as a claim of measured FLOP savings. Second, it isolates the call-count effect only, whereas the realised throughput also depends on how well the expansion overlaps the remaining target layers, which a count-based model cannot capture. Third, it inherits the greedy-decoding assumption of Section 3.3; under stochastic sampling the counts become random variables and Eq. 7 holds in expectation.

D.3 Correction Trigger Rates

Table 7 reports how often the correction mechanism actually fires, measured as the proportion of verification rounds in which a correction branch rather than the original draft carries the step. The counts are dataset-level, obtained by summing the per-sample counters over the whole evaluation split, and they come from the same configuration that produces the main results of Section 4. PEARL cannot trigger a correction at all, so its row is zero by construction and is shown only as a reference point.

The trigger rate separates the three drafting regimes cleanly and in the expected direction: the weakest drafter (Llama3.2-1B) needs a correction in roughly a quarter to a third of all rounds, the Llama2-7B draft in about a fifth, and the well-matched Llama3.1-8B draft in only about one round in eight. This is the same ordering that the trigger rate predicts from Section 4, and it also explains why the throughput margin over PEARL is largest for the weakest drafter: CWV only has an opportunity to act where the draft would otherwise be discarded.

D.4 Memory Overhead Analysis of the Custom Attention Mask

A potential concern with the custom attention mask is whether the generation of multiple branches increases GPU memory usage or access bottlenecks, particularly given that LLM inference is often memory-bound. Our analysis confirms that the custom attention mask does not significantly exacerbate memory-bound constraints, as the additional overhead is marginal compared to the baseline decoding costs (loading model weights). We provide a quantitative breakdown for the Llama2-7B model in Table 8. The memory bandwidth during token generation is dominated by the transfer of model weights (approx. 14 GB). While parallel branch expansion introduces extra tokens into the

Table 7: Correction trigger rate: the share of verification rounds carried by a correction branch instead of the original draft, computed from dataset-level sums of the per-round counters. `early` is the number of such rounds, `draft` the rounds carried by the unmodified draft, and `reject` the rounds in which neither survived and the target resampled. PEARL has no correction mechanism, so its trigger rate is identically zero.

Target–draft pair	Dataset	early	draft	reject	Trigger rate
Llama3.1-70B Llama3.2-1B	GSM8K	881	1718	515	28.3%
	HumanEval	1220	2922	637	25.5%
	MATH500	2157	4264	1393	27.6%
	AIME 2024	994	1494	635	31.8%
	AIME 2025	968	1482	637	31.4%
	MBPP	901	1121	488	35.9%
Llama3.1-70B Llama3.1-8B	GSM8K	780	5659	656	11.0%
	HumanEval	542	4207	309	10.7%
	MATH500	1486	9049	1006	12.9%
	AIME 2024	556	2375	404	16.7%
	AIME 2025	738	3370	424	16.3%
	MBPP	663	2859	372	17.0%
Llama2-70B Llama2-7B	GSM8K	812	3059	619	18.1%
	HumanEval	1954	7471	1417	18.0%
	MATH500	1165	3082	1058	22.0%
	AIME 2024	299	1075	300	17.9%
	AIME 2025	365	914	348	22.4%
	MBPP	2379	6768	2038	21.3%
Llama3.1-70B / PEARL	all	0	–	–	0%

KV cache, the aggregated size of these tokens is negligible (e.g., 100 tokens $\approx$ 52.4 MB) compared to the 14 GB baseline. Furthermore, once a verification step is completed, only the KV cache of accepted tokens is retained, ensuring no long-term memory accumulation.

Table 8: Memory usage breakdown for Llama2-7B during inference. The overhead from parallel branch expansion (additional KV cache) is minimal compared to the static cost of loading model weights.

Component	Size Calculation	Memory Usage
Model Weights	7B params $\times$ 2 bytes	$\sim$14 GB
KV Cache (per token)	32 layers $\times$ (4096 Key + 4096 Value) $\times$ 2 bytes	$\sim$0.524 MB

D.5 Performance on Smaller Scale Models (13B)

To validate the scalability of CWV beyond the 70B parameter scale, we conducted additional experiments using the Vicuna-13B model as the target model and Vicuna-68M as the draft model. The experiments were performed on two NVIDIA H800 GPUs with top-$k = 4$. The results are summarized in Table 9. CWV demonstrates consistent effectiveness on the 13B scale, outperforming both vanilla Speculative Decoding and PEARL. This indicates that the proposed method is not limited to massive 70B models but remains applicable and effective at smaller scales, provided a suitable draft model is available.

For models within the same series (e.g., Llama 3), manufacturers typically release a 7B model as the smallest variant. It is challenging to find suitable, smaller-scale than official 7B models to serve as draft models. Officially released smaller models, trained on similar data and with comparable methodologies, exhibit excellent consistency with their larger counterparts. This strong alignment

Table 9: Experiment results on GSM8K and HumanEval using Vicuna-13B (Target) and Vicuna-68M (Draft). Speed is measured in tokens/s.

	GSM8K		HumanEval	
Method	**Speed**	**Speedup**	**Speed**	**Speedup**
Autoregression	50.60	1.00×	50.60	1.00×
Speculative Decoding	65.82	1.30×	69.05	1.36×
PEARL	76.76	1.52×	76.22	1.51×
CWV	**78.17**	**1.54**×	**78.77**	**1.56**×

leads to remarkable speedups in speculative sampling, as evidenced by the outstanding performance of the Llama3.2-1B and Llama3.1-70B pairing in Table 1 and Table 2 of our paper.

In contrast, smaller models trained by third parties often show suboptimal consistency due to differing training data. Their performance, not only in terms of alignment but also in practical accuracy, frequently falls short of expectations. Consequently, pairing a 7B target model with a third-party small model yields less than ideal results. Even if CWV corrects the current incorrect draft tokens, the draft model's subsequent token predictions remain unreliable, necessitating frequent interventions from the trainable decoder layer. This is not an ideal scenario for showcasing the method's potential.

We would like to emphasize that this performance limitation stems primarily from the unavailability of a suitable draft model, rather than from a lack of generalizability in CWV itself. As demonstrated by Equation 7, CWV reduces verification cost when intermediate predictions successfully seed valid correction branches. We believe that with more appropriately paired models, the benefits of CWV would be even more pronounced.

In addition, we also considered this model pairing early in our research and performed a preliminary analysis. However, we found that the inference speed difference between these two models is minimal—less than a 2x factor, which makes it not worth correcting wrong draft tokens in parallel speculative decoding. Unlike Eagle's draft model which features only a single decoder layer, the Llama3.2-1B architecture incorporates 16 decoder layers. Despite their comparable parameter sizes, the Llama3.2-1B with its increased number of decoder layers demonstrates less impressive performance in inference speed. For this reason, we concluded that this pairing would not serve as a representative or informative test case for our method's capabilities and thus did not proceed with a full set of experiments.

D.6 Challenges with 7B Target Models

We also explored applying CWV to 7B-scale target models. However, identifying suitable draft models smaller than 7B presents a significant challenge. For model series like Llama 3, the 7B (or 8B) variant is typically the smallest official release. As shown in our main results (Table 1), officially released smaller models (e.g., Llama 3.2-1B paired with Llama 3.1-70B) exhibit excellent alignment due to consistent training data and methodologies, leading to remarkable speedups. In contrast, applying speculative decoding to a 7B target often requires using third-party trained draft models (e.g., 100M-300M parameters). These models frequently exhibit suboptimal consistency with the target model due to differing training corpora and alignment strategies. Our preliminary tests showed that such inconsistencies force the trainable decoder layer to intervene frequently, and even when CWV rectifies a token, the draft model's subsequent predictions remain unreliable. This results in diminishing returns for acceleration. Therefore, the limitation in applying CWV to 7B targets stems primarily from the unavailability of high-quality, aligned draft models rather than a lack of generalizability in the CWV methodology itself.

D.7 ACCELERATED DECODING ALGORITHM

Algorithm 1 Correct-While-Verifying Decoding

Input: Target model $\mathcal{M}_t$, Draft model $\mathcal{M}_d$, initial sequence $x_{<t}$, draft length γ, candidates k.
Output: Generated sequence x.
$v_{\text{start}} \leftarrow \text{len}(x)$
$v_{\text{end}} \leftarrow v_{\text{start}} + 1$
$x \leftarrow x_{<t}$
while not stop_condition() **do**
  # Stage 1: Parallel draft and intermediate target prediction
  ASYNC: $\tilde{X} \leftarrow \mathcal{M}_d.\text{generate}(x, \gamma_1)$
  ASYNC: $H, L_{\text{early}} \leftarrow \mathcal{M}_t.\text{early_exit}(x)$
  SYNC
  # Stage 2: Parallel correction expansion and final verification
  $\mathcal{C} \leftarrow \text{Top-k}(L_{\text{early}}, k)$ {get top-k from L_{early}}
  $x_{\text{branch}} \leftarrow \text{ConstructBranches}(x, \tilde{X}, \mathcal{C})$
  $M_{\text{branch}} \leftarrow \text{CustomAttentionMask}(x_{\text{branch}})$
  ASYNC: $\tilde{X}_{\text{corr}} \leftarrow \mathcal{M}_d.\text{generate}(x_{\text{branch}}, M_{\text{branch}}, \gamma_2)$
  ASYNC: $L_{\text{t}} \leftarrow \mathcal{M}_t.\text{verify}(H)$
  SYNC
  {Verify all paths: original draft (index 0) and corrected branches}
  $P, l_{\text{acc}} \leftarrow \text{SelectBestPath}(\tilde{X}_d, \tilde{X}_{\text{corr}}, L_t)$
  $X^{\text{best}} \leftarrow \text{GetPathTokens}(P, \tilde{X}_d, \tilde{X}_{\text{corr}})$
  {Commit accepted tokens from the verified window}
  $x \leftarrow x[: v_{\text{start}}] \oplus X^{\text{best}}[: l_{\text{acc}}]$
  if $P = 0$ **and** $l_{\text{acc}} < (v_{\text{end}} - v_{\text{start}})$ **then**
    # Scenario: Rejection (draft and correct failed)
    $t_{\text{sample}} \leftarrow \text{Sample}(L_t[l_{\text{acc}}])$
    $x \leftarrow x \oplus t_{\text{sample}}$
    $v_{\text{start}} \leftarrow \text{len}(x)$
    $v_{\text{end}} \leftarrow v_{\text{start}} + 1$ {Force short step or restart}
  else
    # Scenario: Success (draft or correct accepted)
    $v_{\text{start}} \leftarrow \text{len}(x)$
    {Append the speculative tail generated in Stage 2 for next round}
    $x \leftarrow x \oplus X^{\text{best}}[l_{\text{acc}} :]$
    $v_{\text{end}} \leftarrow \text{len}(x)$
  end if
end while
return x