

Direct Multi-Token Decoding

Xuan Luo, Weizhi Wang, Xifeng Yan

Department of Computer Science, UC Santa Barbara
{xuan_luo, weizhiwang, xyan}@cs.ucsb.edu

Abstract

Recent studies suggest that pre-trained large language models (LLMs) might develop distinct functional roles across their layers: early layers focus on understanding the input context, middle layers handle task-specific processing, and late layers map abstract representations to output tokens. Inspired by the intuition that humans can produce an extended sequence of speech from a single act of reading and thinking, we hypothesize that a single pass through the early and middle layers could provide sufficient information to support the decoding of multiple future tokens. Accordingly, we propose Direct Multi-Token Decoding (DMTD), which performs only one full forward pass for every n tokens and reuses the late layers to decode multiple subsequent tokens. Unlike speculative decoding, our method introduces no additional computation, auxiliary routines, or post-generation verification. With minimal training overhead, a Qwen3-4B model fine-tuned with DMTD achieves up to a $2.15\times$ speedup with marginal performance loss, and our scaling analysis indicates that its performance could be further improved with larger-scale training.

Code  [luoxuan-cs/Direct-Multitoken-Decoding](https://github.com/luoxuan-cs/Direct-Multitoken-Decoding)
Model  [xuan-luo/DMTD-Qwen3-4B](https://huggingface.co/xuan-luo/DMTD-Qwen3-4B)

1 Introduction

Transformers are the default choice for building large language models (LLMs). The original Transformer (Vaswani et al., 2017) employed an encoder-decoder structure for sequence-to-sequence modeling, where the encoder processes input sequences for natural language understanding (NLU) and the decoder produces outputs for natural language generation (NLG). Subsequently, decoder-only architectures (Brown et al., 2020; Touvron et al., 2023; Guo et al., 2025) have become mainstream due to their scalability and simplicity. By leveraging masked self-attention, these decoder-only models process sequences causally, enabling efficient parallel training and supporting versatile multi-tasking.

Recent studies reveal that pre-trained decoder-only transformers may exhibit specialized functional roles across their layers (Sun et al., 2025; Ferrando et al., 2024; Merullo et al., 2024; Skean et al., 2025). Specifically, these layers can be categorized into three functional stages. First, early layers encode syntactic and semantic features of the input context (Hu et al., 2024; Azaria & Mitchell, 2023). Next, middle layers handle reasoning and task-specific processing (Luo et al., 2025; Wang et al., 2025). Finally, late layers map latent representations to discrete tokens (Chuang et al., 2024; Zhang et al., 2024). This layered specialization suggests that, while encoder-decoder architectures explicitly define encoding and decoding components, decoder-only models might implicitly develop a similar structure through training. To reflect their roles, we refer to these layers as encoding, thinking, and decoding layers, respectively, as illustrated in Figure 1 (left). Notably, these functional stages are not defined by sharp boundaries; instead, they form a functional continuum where the representations undergo a gradual phase transition across the model’s depth. For our implementations, we empirically set the layer boundaries between these stages.

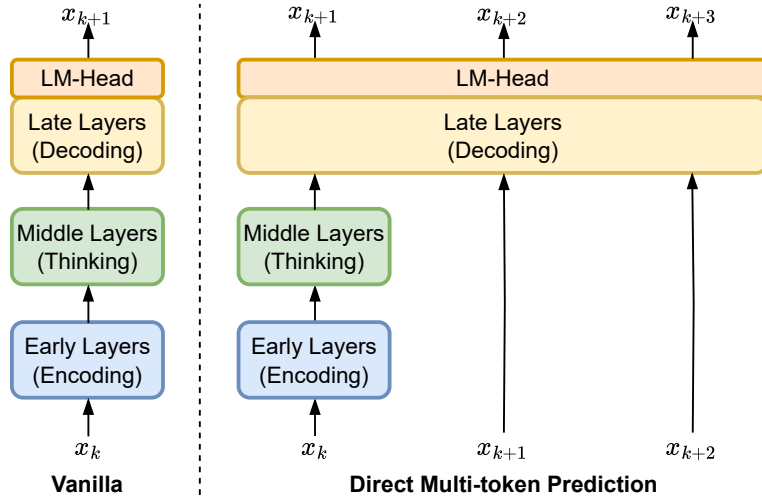


Figure 1: Vanilla next token prediction vs. direct multi-token decoding.

This three-stage functional organization mirrors the cognitive sequence of human communication, where reading and thinking serve as the precursors to writing. However, a fundamental discrepancy in efficiency remains: while humans can produce an extended sequence of words from a single act of reading and thinking, current LLMs must re-traverse the entire encoding and thinking layers for every individual token. This leads us to wonder: could LLMs similarly execute these layers only once to support the decoding of multiple future tokens? This hypothesis is supported by layer-skipping studies (Luo et al., 2025; Zeng et al., 2023; Tan et al., 2024), which indicate that computational redundancy exists within the transformer’s pipeline. The presence of such redundancy suggests that the encoding and thinking layers can be tasked with more than just predicting the next token. Instead of simply discarding this redundancy, we could leverage it to encapsulate more future information into hidden states, thereby enabling efficient multi-token decoding.

In this work, we propose Direct Multi-Token Decoding (DMTD), which reuses the late layers to directly decode multiple tokens. Unlike the vanilla decoder-only transformer that generates tokens one by one through full forward passes, the proposed DMTD operates in fixed multi-token cycles. Figure 1 (right) demonstrates the generation pipeline of DMTD in a single cycle. DMTD performs only one full forward pass at the beginning of the cycle and then reuses the late layers to decode multiple tokens consecutively. This cyclic configuration transforms the irregular computational redundancy observed in pre-trained LLMs (Luo et al., 2025; Zeng et al., 2023) into a structured, periodic pattern that enables efficient multi-token decoding. Unlike existing multi-token prediction methods, DMTD introduces no extra computation (Li et al., 2024a;b; 2025; Liu et al., 2024), auxiliary LM heads (Cai et al., 2024), or post-generation verification like speculative decoding (Leviathan et al., 2023). Nevertheless, because the directly decoded tokens naturally serve as drafts, DMTD remains fully compatible with speculative decoding, where these tokens can be further verified against the full model to guarantee identical output distribution.

We evaluated Direct Multi-Token Decoding (DMTD) across various benchmarks. By repurposing the final 8 layers of the 36-layer Qwen3-4B (Yang et al., 2025) to decode 4 tokens per cycle, DMTD preserves 95.9% of the original performance while achieving up to a $2.15\times$ inference speedup. Furthermore, our observations indicate that DMTD’s effectiveness scales favorably with model size and training data, suggesting a promising trajectory for larger architectures and more extensive training. Together, these results point to a new perspective on efficient decoding: rather than making each forward pass faster, DMTD reduces how often the model needs to read and think at all—a gain that comes not from speculating the model’s outputs, but from making fuller use of what it has already computed.

2 Method

In this section, we present the training and inference processes of the proposed direct multi-token decoding (DMTD). During training, we use a cyclical masking strategy to enable efficient learning of multiple future tokens. During inference, decoding proceeds sequentially across cycles, incorporating a cyclical KV refilling mechanism to recover missing KV-cache entries.

2.1 Parallel Training with Cyclical Masking

We propose a cyclical masking strategy to unify multi-token predictions within a single sequence during training. In standard next-token prediction, models learn to forecast one token at a time based on the preceding sequence. For multi-token prediction, our approach extends this by enabling the model to learn multiple future tokens simultaneously, all from the same input sequence. It does so by masking specific parts of the sequence intentionally, which directs the model to focus on predicting different future positions without needing separate sequences (Li et al., 2024a; Cai et al., 2024; Liu et al., 2024). We define the cycle length of multi-token decoding as τ . Figure 2 illustrates the training pipeline for $\tau = 3$. Given an input sequence $\mathbf{x} = x_0, x_1, \dots, x_n$ and cycle length τ , the training process consists of three phases:

Encoding Layers: We first obtain the initial token embeddings from the input sequence using the embedding layer: $\mathbf{h}_{\text{emb}} = \text{Embed}(\mathbf{x})$. These embeddings are then processed by the encoding layers to produce the encoding representations $\mathbf{h}_{\text{enc}}$:

$$\mathbf{h}_{\text{enc}} = \text{EncodingLayers}(\mathbf{h}_{\text{emb}}). \quad (1)$$

Thinking Layers: The encoding representations $\mathbf{h}_{\text{enc}}$ are further refined through the thinking layers to generate thinking representations $\mathbf{h}_{\text{think}}$:

$$\mathbf{h}_{\text{think}} = \text{ThinkingLayers}(\mathbf{h}_{\text{enc}}). \quad (2)$$

Decoding Layers with Masking: Our training approach uses a masking strategy to simulate different execution paths within a single forward pass. A mask is applied based on the cycle length τ to selectively combine the input embeddings $\mathbf{h}_{\text{emb}}$ and thinking representations $\mathbf{h}_{\text{think}}$. For position indices $\mathbf{p} = \{0, 1, \dots, n-1\}$, we create a binary pattern mask $\mathbf{M}$ where:

$$m_i = \begin{cases} 1 & \text{if } p_i \bmod \tau = 0, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

For example, with a cycle length of $\tau = 3$, the masking pattern becomes $[1, 0, 0, 1, 0, 0, 1, 0, 0, \dots]$. The masked hidden states are then computed as follows:

$$\mathbf{h}_{\text{masked}} = \mathbf{h}_{\text{emb}} + \mathbf{h}_{\text{think}} \odot \mathbf{M}. \quad (4)$$

Alternatively, we can also leverage the encoding representations $\mathbf{h}_{\text{enc}}$ for multi-token decoding by using $\mathbf{h}_{\text{enc}}$ instead of $\mathbf{h}_{\text{emb}}$ to compute $\mathbf{h}_{\text{masked}}$. Under this setting, we reuse the encoding layers as well as the decoding layers for multi-token decoding.

The resulting masked hidden states $\mathbf{h}_{\text{masked}}$ are then processed through the decoding layers and the LM head to obtain the output logits $\mathbf{z}$:

$$\mathbf{z} = \text{LMHead}(\text{DecodingLayers}(\mathbf{h}_{\text{masked}})). \quad (5)$$

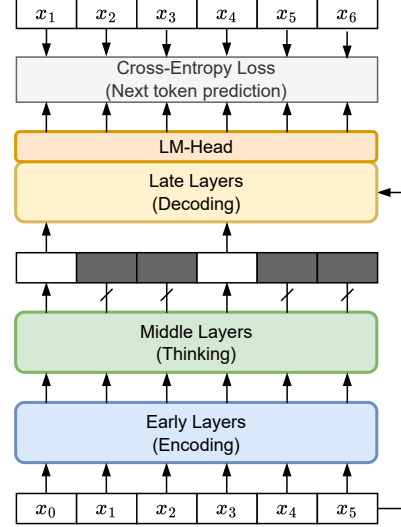


Figure 2: DMTD training pipeline with a cycle length of 3.

	Prefilling Stage					Decoding Stage					
Late Layers	1	1	1	1	1	2	3	4	5	6	7
Middle Layers	1	1	1	1	1	4	4	4	7	7	7
Early Layers	1	1	1	1	1	4	4	4	7	7	7
	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}

Figure 3: Cyclical refilling with a cycle length of 3. There are three blocks within each column, representing the early, middle, and late layers. Blocks with the same color are computed in the same forward pass. The numbers on the blocks represent the index of the forward pass.

Finally, we will simply use the vanilla next token prediction loss for optimization:

$$\mathcal{L} = \frac{1}{n} \sum_{i=0}^{n-1} \text{CrossEntropy}(\mathbf{z}_i, x_{i+1}). \quad (6)$$

Although we use the vanilla next-token prediction loss, the masking strategy enables the model to learn predictions for multiple future tokens. This differs from methods like Medusa (Cai et al., 2024) or DeepSeek-V3 (Liu et al., 2024), which rely on multiple execution paths for multi-token prediction: each path has its own module and its own cross-entropy loss against targets shifted further into the future. DMTD instead simply learns all n predictions within a single sequence.

2.2 Multi-token decoding with cyclical KV refilling

The proposed DMTD performs decoding in fixed multi-token cycles, leveraging the specialized roles of the late layers to generate multiple tokens efficiently. In each cycle, the first forward pass processes the input through all layers, while subsequent forward passes within the cycle use only the late layers. However, as generation progresses, skipping the early and middle layers results in missing entries in the key-value cache (KV-cache) (Pope et al., 2023), which stores intermediate representations the attention module needs for subsequent generation. These missing KV cache entries can degrade the quality of new tokens due to incomplete context. To address this, we introduce a cyclical KV refilling strategy that restores missing KV cache entries from previous cycles.

Figure 3 outlines the decoding process of our method. Consider an input context of tokens x_0, x_1, x_2, x_3, x_4 . The process begins by forwarding all the input context through the early, middle, and late layers for prefilling, which also generates the first output token x_5 . In the subsequent decoding stage of this cycle, the model forwards x_5 through only the late layers to produce x_6 , and then forwards x_6 through the late layers to generate x_7 . This completes the first cycle, which involves three forward passes in total. At the start of the second cycle, our model forwards x_5, x_6, x_7 together through the early and middle layers to refill the KV cache for x_5 and x_6 , while only x_7 is processed through the late layers to generate x_8 . This refilling mechanism ensures that the full context remains available for generation, eliminating the need for speculative decoding methods to mitigate error propagation.

It can be observed that the overall computational load of the proposed DMTD is the same as the vanilla transformer. Therefore, how can our method achieve faster inference? The reason lies in the memory-bound nature of large language model (LLM) inference (Leviathan et al., 2023; Li et al., 2024a; Cai et al., 2024). In this scenario, GPU computational resources are underutilized, and inference speed depends primarily on the number of modules processed rather than the total computational volume (Recasens et al., 2025). Speculative decoding exploits the same property, but it does so by spending additional computation on drafting and verification in exchange for fewer sequential forward passes (Leviathan et al., 2023). DMTD requires no such trade: by processing fewer layers per token, it accelerates inference while keeping the computational load identical to that of a vanilla transformer.

Model	ARC-E	ARC-C	WinoGrande	GSM8K	CoQA	MBPP+	Overall
Vanilla	0.934	0.922	0.657	0.907	0.805	0.669	100.0%
C2	0.930	0.897	0.701	0.901	0.798	0.693	100.9%
C3	0.921	0.886	0.673	0.889	0.780	0.698	99.4%
C4	0.916	0.881	0.652	0.866	0.749	0.627	95.9%
C6	0.872	0.801	0.601	0.500	0.672	0.536	81.7%

Table 1: The influence of different cycle lengths. $C\tau$ denotes a decoding cycle of τ tokens. The overall score reflects the average relative performance compared to the vanilla Qwen3-4B (Yang et al., 2025).

3 Experiments

3.1 Implementation details

We implement DMTD using the pre-trained Qwen3-4B (36 layers) (Yang et al., 2025), reusing the final 8 layers for multi-token decoding with a default cycle length of $\tau = 3$. The model is fine-tuned on the AM-Thinking-v1-Distilled dataset (Tian et al., 2025) for 1 epoch. Training is conducted using the AdamW optimizer with a learning rate of 1×10^{-4} , $\beta_1 = 0.9$, and $\beta_2 = 0.95$. We employ a cosine learning rate scheduler with a 0.1 warmup ratio and a global batch size of 128. Due to resource constraints, both training and evaluation are conducted with a maximum context length of 4096 tokens.

We assess our method on ARC-Easy, ARC-Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2021), GSM8K (Cobbe et al., 2021), CoQA (Reddy et al., 2019), and MBPP+ (Liu et al., 2023). ARC (Clark et al., 2018) examines knowledge and reasoning through grade-school science questions. WinoGrande (Sakaguchi et al., 2021) tests commonsense reasoning with adversarial Winograd schema challenges. GSM8K (Cobbe et al., 2021) evaluates multi-step mathematical reasoning using grade-school word problems. CoQA (Reddy et al., 2019) measures conversational question-answering skills, including coreference and pragmatic reasoning. MBPP+ (Liu et al., 2023) is an enhanced variant of MBPP that evaluates code generation on programming tasks. We apply 4-shot prompting for GSM8K, while the others are evaluated in a zero-shot setting. We utilize the think mode (Yang et al., 2025) with chain-of-thought (CoT) prompting (Wei et al., 2022) for all benchmarks except CoQA, which does not require deep reasoning. All tasks involve continuous generation to assess the multi-token decoding capability of the proposed method.

3.2 Decoding cycle length

We evaluate the performance of DMTD across various cycle lengths using a default setup with 8 decoding layers. Table 1 shows the results, with each model using a consistent cycle length for both training and evaluation. All generations are performed directly without post-verification.

As shown in Table 1, our proposed method performs effectively for cycle lengths up to 4, with performance gradually declining as the cycle length increases. However, performance noticeably drops beyond this point, falling to 81.7% at a cycle length of 6. We hypothesize that this decline results from the limited dimensionality of the hidden states, which restricts their capacity to capture sufficient information about future tokens, thus hindering effective long-range multi-token generation. Full-scale pre-training of the proposed method on larger models could potentially support longer prediction horizons.

3.3 Impact of encoding and decoding layer allocation

We investigate the relative importance of encoding and decoding layers in DMTD by evaluating three reuse budgets: 4, 8, and 16 layers. All experiments are conducted with a cycle length of $\tau = 3$. As shown in Table 2, the results indicate that decoding layers are essential for effective multi-token decoding. Configurations reusing only encoding layers (ExD0) yield suboptimal performance, reaching only 77.9% even with a 16-layer

Setting	ARC-E	ARC-C	WinoGrande	GSM8K	CoQA	MBPP+	Overall
Vanilla	0.934	0.922	0.657	0.907	0.805	0.669	100.0%
Reuse 4 Layers							
E4D0	0.412	0.364	0.517	0.048	0.532	0.005	39.1%
E2D2	0.919	0.878	0.663	0.878	0.793	0.645	97.7%
E0D4	0.918	0.882	0.665	0.889	0.758	0.638	97.1%
Reuse 8 Layers							
E8D0	0.540	0.470	0.497	0.194	0.604	0.061	48.3%
E4D4	0.922	0.876	0.670	0.890	0.808	0.683	99.4%
E0D8	0.921	0.886	0.673	0.889	0.780	0.698	99.4%
Reuse 16 Layers							
E16D0	0.741	0.609	0.535	0.544	0.717	0.614	77.9%
E8D8	0.921	0.890	0.683	0.898	0.812	0.679	100.1%
E0D16	0.928	0.898	0.685	0.907	0.802	0.685	100.5%

Table 2: Impact of layer allocation on performance. ExDy denotes a configuration with x encoding layers and y decoding layers reused.

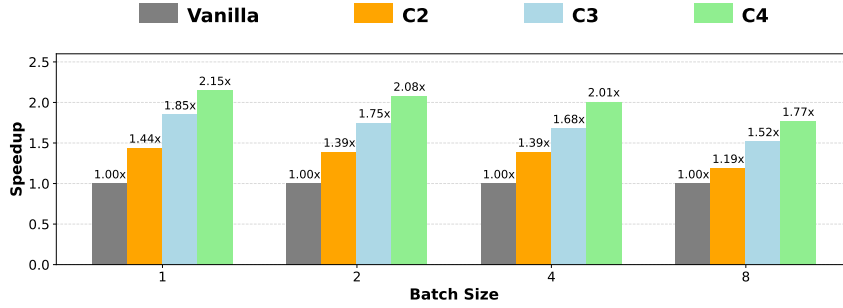


Figure 4: Inference speedup of DMTD (E0D8) over vanilla Qwen3-4B on a single A100-40GB GPU, across batch sizes and cycle lengths, with a fixed generation length of 1024 tokens.

budget. Increasing the encoding capacity alone cannot compensate for the absence of late-stage decoding layers in multi-token prediction. In contrast, prioritizing decoding layers (E0Dy) consistently improves performance. Furthermore, balanced allocations (e.g., E8D8) achieve results comparable to purely decoding-focused ones, suggesting that both layer types contribute effectively once a sufficient decoding foundation is established.

3.4 Inference speedup

To evaluate inference efficiency, we compare the default DMTD configuration (reusing the last 8 layers) against the vanilla Qwen3-4B baseline. All measurements are conducted on a single NVIDIA A100-40GB GPU using a static input and generation length of 1024 tokens. As illustrated in Figure 4, our method achieves substantial throughput gains that scale with the cycle length. For instance, with a cycle length of 4, DMTD provides a $2.15\times$ speedup at a batch size of 1. Notably, because all samples follow identical execution paths, this synchronous configuration is more robust to batch-size scaling than asynchronous approaches. Even as the system transitions toward a compute-bound state at larger batch sizes, DMTD still maintains a $1.77\times$ speedup at batch size 8.

3.5 Scaling analysis

We first investigate the scaling behavior of DMTD as a function of training data volume and model size, using the default E0D8 configuration with a cycle length of 3 ($\tau = 3$). As illustrated in Figure 5, there is a consistent decrease in cross-entropy loss as the volume of training data increases across all model sizes (Qwen3-0.6B, 1.7B, and 4B). The x-axis represents the number of training tokens (in billions) on a logarithmic scale, while the y-axis

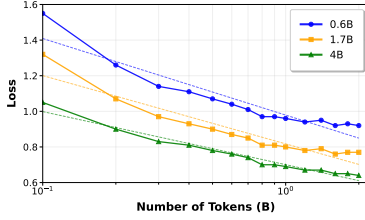


Figure 5: Scaling law of DMTD during training.

	ARC	Wino.	GSM8K	CoQA	MBPP+	Overall
Qwen3-0.6B						
Vanilla	0.750	0.499	0.751	0.706	0.466	100%
Ours	0.625	0.489	0.532	0.550	0.296	79.4%
Qwen3-1.7B						
Vanilla	0.881	0.566	0.828	0.776	0.598	100%
Ours	0.814	0.540	0.716	0.715	0.473	89.6%
Qwen3-4B						
Vanilla	0.928	0.657	0.907	0.805	0.669	100%
Ours	0.904	0.673	0.889	0.780	0.698	99.4%

Table 3: DMTD performance across different model sizes.

shows the cross-entropy loss. These trends closely approximate log-linear relationships, suggesting that the multi-token prediction capability of DMTD scales predictably. While our current experiments are limited to fine-tuning on a 1.5B token dataset, these scaling curves indicate that large-scale continued pre-training, followed by post-training alignment, could further unlock the potential of this architecture.

The impact of model capacity on downstream task performance is further detailed in Table 3. We observe that larger models benefit more from our method. Specifically, the Qwen3-4B model maintains 99.4% of the vanilla performance, whereas the smaller Qwen3-0.6B model retains only 79.4%. We hypothesize that the increased parameter count and higher dimensionality of larger models provide the necessary representational capacity to encapsulate rich anticipatory information for multiple future tokens. These results suggest that DMTD might be more suited for large-scale LLMs.

3.6 Compatibility with speculative decoding

Speculative decoding (Leviathan et al., 2023) accelerates autoregressive generation by introducing a lightweight draft model that efficiently produces multiple tokens. These draft tokens are then verified against a slower target model in a single forward pass: tokens consistent with the target distribution are accepted, while rejected ones are resampled. The procedure preserves the target model’s output distribution exactly, so the generated text matches vanilla decoding while requiring fewer full forward passes. This could provide essential acceleration in a memory-bound decoding scenario.

In this subsection, we demonstrate that the proposed DMTD is compatible with speculative decoding and can operate as a self-speculative decoder, serving as both the draft model and the target model. The late layers of DMTD naturally act as the draft model: tokens produced by forwarding only through these layers can be treated as draft tokens. During cyclical KV refilling, the previous cycle’s tokens are forwarded through the early and middle layers; continuing this pass through the late layers and the lm-head as well yields the target distribution at every drafted position. The target distribution is then used to verify the draft tokens, accepting those that match and rejecting the rest.

Beyond this basic draft-then-verify pipeline, how the draft tokens are organized and how verification is carried out admit several designs. The simplest is linear verification (Leviathan et al., 2023), which checks a single chain of draft tokens and stops at the first rejection. Tree verification (Li et al., 2024a) instead keeps several candidate continuations under a shared attention mask, so a rejection need not discard the entire block and the longest accepted path can be selected. Medusa-style verification (Cai et al., 2024) further builds a fixed sparse tree from the top- k candidates at each drafted position and replaces rejection sampling with a typical acceptance criterion. As shown in Table 4, we incorporate these methods into DMTD for speculative decoding and evaluate the resulting speedups under an inference batch size of 1. Although these methods work well with DMTD, we emphasize that speculative decoding is not the intended use of DMTD; the results are reported only to

Method	GSM8K	HumanEval(2021)	MT-Bench(2023)
Vanilla	1.00×	1.00×	1.00×
DMTD	2.07×	2.05×	2.05×
DMTD + linear SD	1.82×	1.81×	1.47×
DMTD + tree SD	1.92×	1.94×	1.64×
DMTD + Medusa	2.06×	1.96×	1.58×

Table 4: Speedup of DMTD with speculative decoding.

Setting	ARC-E	ARC-C	WinoGrande	GSM8K	CoQA	MBPP+	Overall
Vanilla	0.934	0.922	0.657	0.907	0.805	0.669	100.0%
DMTD	0.921	0.886	0.673	0.889	0.780	0.698	99.4%
Frozen-E&T	0.326	0.300	0.388	0.023	0.774	0.005	37.7%

Table 5: Performance comparison between DMTD and the Frozen-E&T baseline, where encoding and thinking layers are kept fixed during training.

show compatibility in cases where the output distribution of the original model must be preserved.

3.7 Is DMTD merely layer skipping? A mechanistic study

While the previous results demonstrate the efficacy of DMTD, the source of its performance gains remains to be clarified. Specifically, we investigate whether the encoding and thinking layers learn to predict anticipatory information about future tokens. Alternatively, the late layers might already possess enough capacity to generate tokens even when initial stages are skipped. In this case, our specialized training would be unnecessary, and DMTD would simply be exploiting the late layers for generation.

To distinguish between these two hypotheses, we conduct a control experiment using a “Frozen-E&T” setting. We freeze the parameters of all encoding and thinking layers and fine-tune only the final 8 decoding layers. Under the E0D8 configuration with a cycle length of 3, the early stages are restricted to their original representations. This forces the decoding layers to attempt multi-token prediction without any specialized anticipatory information from the preceding layers. As summarized in Table 5, the Frozen-E&T setting results in a catastrophic performance collapse on nearly all benchmarks, particularly those requiring complex reasoning. CoQA remains the sole exception as it only involves short-form generation that does not require complex reasoning. This confirms that the encoding and thinking layers learn to encapsulate future information to enable multi-token decoding.

3.8 Generalization on different models

To verify that DMTD is not specific to the Qwen3 architecture, we extend our method to Llama-3.1-8B-Instruct, Llama-3.2-3B-Instruct (Grattafiori et al., 2024), and OLMo-2-1124-7B-Instruct (OLMo et al., 2024). For each model, we adopt the same E0D8 configuration with a cycle length of 4, fine-tuning on the OpenPerfectBlend dataset (Xu et al., 2024) dataset. As summarized in Table 6, DMTD generalizes consistently across all three models, retaining 97.5%, 95.6%, and 96.0% of vanilla performance on average, respectively. These results demonstrate that DMTD is compatible with other decoder-only architectures.

3.9 Adaptive multi-token decoding

We extend DMTD with an adaptive mechanism to dynamically determine layer skipping within each cycle. While the first token of each cycle always undergoes a full forward pass, subsequent tokens are routed based on a linear projection of their hidden states followed by a sigmoid activation. If the resulting score exceeds 0.5, the model executes the encoding and thinking layers; otherwise, it skips directly to the decoding layers. To encourage

	ARC-E	ARC-C	Wino.	GSM8K	CoQA	MBPP+	Overall
Llama-3.1-8B-Instruct							
Vanilla	0.947	0.869	0.646	0.843	0.780	0.725	100%
Ours	0.920	0.842	0.647	0.825	0.782	0.672	97.5%
Llama-3.2-3B-Instruct							
Vanilla	0.903	0.806	0.579	0.775	0.792	0.511	100%
Ours	0.867	0.761	0.572	0.749	0.774	0.460	95.6%
OLMo-2-7B-Instruct							
Vanilla	0.887	0.759	0.563	0.770	0.794	0.474	100%
Ours	0.874	0.726	0.558	0.733	0.780	0.423	96.0%

Table 6: DMTD generalization across models, trained on OpenPerfectBlend with a cycle length of 4.

Setting	ARC-E	ARC-C	WinoGrande	GSM8K	CoQA	MBPP+	Overall
Vanilla	0.934	0.922	0.657	0.907	0.805	0.669	100.0%
C2	0.930	0.897	0.701	0.901	0.798	0.693	100.9%
C4	0.916	0.881	0.652	0.866	0.749	0.627	95.9%
Adaptive	0.932	0.899	0.706	0.902	0.801	0.678	100.8%

Table 7: Performance of adaptive multi-token decoding compared to fixed cycle lengths.

computational efficiency, we apply an L_1 regularization penalty (1×10^{-3}) to the sigmoid scores during training, biasing the model toward skipping.

Under the *E0D8* configuration with a training cycle length of 4, the adaptive router results in approximately 50.8% of tokens undergoing a full forward pass. This corresponds to an effective average cycle length of 2. As shown in Table 7, this dynamic approach outperforms the fixed $\tau = 4$ baseline and achieves performance comparable to the fixed $\tau = 2$ setting, which shares a similar 50% full-pass ratio. These results suggest that the model can learn to allocate computational resources on a per-token basis, leveraging the encoding and thinking layers only when necessary to maintain generation quality.

4 Related works

4.1 Large language models

Initially, the Transformer architecture (Vaswani et al., 2017) adopted an encoder-decoder structure for sequence-to-sequence modeling, where the encoder uses bidirectional attention to understand the input context, and the decoder applies causal attention for token generation. In contrast, decoder-only models, such as the GPT series (Brown et al., 2020; Ouyang et al., 2022), emerged to prioritize generative tasks by relying solely on causal self-attention. Recent studies reveal that decoder-only models implicitly develop a three-layer functional specialization during training, mirroring an encoding-thinking-decoding pipeline. Early layers focus on syntactic and semantic encoding, transforming raw inputs into a stable embedding space critical for contextual understanding (Sun et al., 2025). Middle layers handle reasoning and task-specific abstraction, compressing information and enabling complex processing, such as multi-step reasoning, with greater robustness to layer manipulation (Skean et al., 2025). Late layers specialize in token-level predictions, refining representations for generation but often discarding broader contextual features (Sun et al., 2025; Skean et al., 2025). Mixture-of-Depths (MoD) () demonstrates that, with large-scale pre-training, we can use a router to skip the redundant layers in transformers. Furthermore, FlexiDepth (Luo et al., 2025; Men et al., 2025) demonstrates that these layers exhibit varying sparsity in a bowl-like pattern. However, they represent irregular skipping patterns, which are difficult to provide acceleration in memory-bound scenarios. These findings inspired our Direct Multi-Token Decoding (DMTD) method, which leverages this layer specialization by

cyclically reusing late layers to efficiently generate multiple tokens, repurposing redundant computations in pre-trained LLMs without additional components.

4.2 Multi-token prediction

To accelerate inference in decoder-only models, recent works have explored multi-token prediction techniques that enable parallel generation of multiple tokens, addressing the memory bottlenecks of autoregressive decoding (Leviathan et al., 2023). Early approaches, such as speculative decoding (Leviathan et al., 2023), leverage smaller draft models to propose multiple candidate tokens in parallel, verifying them against the target LLM to achieve speedups without altering output distributions. Building on this, methods like MTP (Gloeckle et al., 2024) and Medusa (Cai et al., 2024) introduce multiple decoding heads on top of the LLM to predict several subsequent tokens simultaneously. The EAGLE series (Li et al., 2024a;b; 2025) rethinks speculative sampling at the feature level, resolving uncertainty in intermediate representations by advancing token sequences. BiTA (Lin et al., 2025) introduces prefix and mask embeddings with bidirectional tuning to enable multi-token generation without a separate draft model, while Speculative Streaming (Bhendawade et al., 2024) attaches lightweight stream embeddings to the target model itself to draft multiple future tokens, eliminating the need for an auxiliary draft model altogether.

More fundamentally, the methods above share a “generate-then-correct” philosophy: an inexpensive, imprecise mechanism proposes candidate tokens, and correctness is restored post-hoc through verification and rejection sampling. DMTD instead explores an orthogonal “think-then-generate” philosophy, where reliability is established upfront rather than corrected afterward. Our preliminary adaptive variant (Section 3.9) takes a first step in this direction by allowing the cycle length to vary rather than remain fixed a priori; however, it still lacks a principled mechanism for the model to explicitly perceive how many tokens it can safely decode in the current cycle. We view this as an early step and hope it inspires more refined mechanisms for adaptive, verification-free multi-token decoding.

5 Limitations

Since the pre-training dataset of Qwen3 is not publicly available (Qwen3 utilizes a large-scale training dataset consisting of approximately 36 trillion tokens. We only used 1.5B tokens), we were unable to conduct full continual training on the complete dataset to assess the performance of a fully developed DMTD. As shown in Figure 5, its performance is expected to further improve with access to additional training data. Overall, DMTD offers a simple paradigm that merits further investigation, particularly in regimes where speculative decoding becomes harder to exploit. Even recent speculative decoding methods such as DFlash (Chen et al., 2026) and DSpark (Cheng et al., 2026) may see reduced efficiency in Mixture-of-Experts (MoE) architectures under large batch sizes and dispersed expert routing (Qian et al., 2026; Kumar et al., 2026). DMTD performs no rollback and introduces no computation beyond that of a vanilla transformer, and all sequences in a batch follow an identical execution path, so it could be a good alternative in this setting. A systematic study of this behavior is left to future work.

6 Conclusion

In this work, we introduce Direct Multi-Token Decoding (DMTD), a paradigm that enables sustained multi-token generation without introducing additional parameters or requiring post-generation verification, as is the case with speculative decoding. Instead, DMTD leverages the inherent redundancy present in pre-trained LLMs and repurposes it into fixed cycles of multi-token generation. Our experimental results demonstrate not only the feasibility of this approach but also suggest that its performance can further improve with larger training datasets, opening a promising new direction for accelerating LLM inference.

Acknowledgements

We would like to thank Facebook (now Meta) for donating the A100-40G GPUs used in our experiments. We also gratefully acknowledge the generous support of the NVIDIA Academic Grant Program and NCSA DeltaAI through allocation CIS250709 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation.

References

- Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models. *arXiv preprint arXiv:2402.11131*, 2024.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Kaplan, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020.
- 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 *Forty-first International Conference on Machine Learning*, 2024.
- Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.
- Yung-Sung Chuang, Yujia Xie, Hongyin Luo, et al. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*, 2024.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. Towards understanding factual knowledge of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- Avinash Kumar, Sujay Sanghavi, and Poulami Das. Test-time speculation. *arXiv preprint arXiv:2605.09329*, 2026.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2023.

- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*, 2024a.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024b.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.
- Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bit: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 2025.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 2023.
- Xuan Luo, Weizhi Wang, and Xifeng Yan. Adaptive layer-skipping in pre-trained LLMs. In *Second Conference on Language Modeling*, 2025.
- Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.
- Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Mishkin, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 2022.
- Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 2023.
- Yu-Yang Qian, Hao-Cong Wu, Chen Chen, Jiacheng Sun, Zhenhua Dong, Peng Zhao, and Zhi-Hua Zhou. Adaflash: Adaptive speculative decoding via on-policy distilled diffusion drafters. *arXiv preprint arXiv:2607.19223*, 2026.
- Pol G Recasens, Ferran Agullo, Yue Zhu, Chen Wang, Eun Kyung Lee, Olivier Tardieu, Jordi Torres, and Josep Ll Berral. Mind the memory gap: Unveiling gpu bottlenecks in large-batch llm inference. *arXiv preprint arXiv:2503.08311*, 2025.
- Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 2019.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.
- Oscar Skean, Md Rifat Arefin, Dan Zhao, et al. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025.

- Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA, 2025*.
- Zhen Tan, Daize Dong, Xinyu Zhao, Jie Peng, Yu Cheng, and Tianlong Chen. DLO: dynamic layer operation for efficient vertical scaling of llms. *CoRR*, 2024.
- Xiaoyu Tian, Yunjie Ji, Haotian Wang, Shuaiting Chen, Sitong Zhao, Yiping Peng, Han Zhao, and Xiangang Li. Not all correct answers are equal: Why your distillation source matters. *arXiv preprint arXiv:2505.14464*, 2025.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, and Lachaux... Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017*.
- Hanyu Wang, Bochuan Cao, Yuanpu Cao, and Jinghui Chen. Truthflow: Truthful LLM generation via representation flow correction. In *Forty-second International Conference on Machine Learning*, 2025.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 2022.
- Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, et al. The perfect blend: Redefining rlhf with mixture of judges. *arXiv preprint arXiv:2409.20370*, 2024.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Dewen Zeng, Nan Du, Tao Wang, Yuanzhong Xu, Tao Lei, Zhifeng Chen, and Claire Cui. Learning to skip for language modeling. *CoRR*, 2023.
- Jianyi Zhang, Da-Cheng Juan, Cyrus Rashtchian, et al. SLED: Self logits evolution decoding for improving factuality in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 2023.