On the Information Loss of Multi-Token Prediction: Origin and Solution

Tangyu Jiang[1] **Zhanke Zhou**[1] **Haodi Wang**[2]
Yiu-ming Cheung[1] **Xiaohua Jia**[2] **Bo Han**[1†]
[1]Department of Computer Science, Hong Kong Baptist University
[2]Department of Computer Science, City University of Hong Kong

Abstract

Multi-token prediction (MTP) accelerates autoregressive decoding through block-wise prediction, but remains vulnerable to error accumulation across sequentially generated blocks. To investigate this limitation, we analyze the distinction between block-wise and sequence-level cumulative information loss (CIL). Specifically, when the model generates using its own previous outputs without access to reference answers, the input-conditional gap equals the cross-block dependence of the reference targets. To improve prediction when these generated prefixes are imperfect, we propose *Dual-MTP*, which supervises shared prediction heads under full and randomly masked contexts. Together, complete-prefix supervision and context perturbations provide complementary training signals while retaining the MTP architecture. To characterize this training objective, we use the normalized dual loss to bound the divergence between the model's predictive distributions under full and masked contexts. Experiments across five backbones and six tasks spanning mathematical reasoning and GLUE understanding show that Dual-MTP achieves the highest average task score on four of five backbones, with gains of up to 2.46 points over MTP. On mathematical reasoning tasks, Dual-MTP also achieves $1.4\times$–$1.9\times$ the decoding throughput of the NTP reference. Together, these results show that dual-context supervision can improve prediction accuracy while retaining efficient multi-token decoding. Code is available at `https://github.com/tmlr-group/dual-mtp`.

1 Introduction

Large Language Models (LLMs) [37, 16, 14] have achieved remarkable success across a wide range of tasks [40, 32, 42, 9]. The standard training paradigm for these models is Next-Token Prediction (NTP), in which each token is predicted conditioned on the preceding tokens available within the model's context window [36, 15, 24]. While effective, NTP provides only a single-token supervision signal at each position, which may limit the richness of the learned representations, particularly for tasks that require multi-step reasoning [48].

Multi-Token Prediction (MTP) has recently emerged as a compelling alternative to the NTP objective [12, 22, 34]. Rather than predicting the next token at each position, MTP partitions the target sequence into blocks of multiple future tokens and trains the model to forecast an entire block at once, providing denser supervision per position. Concretely, MTP employs a shared backbone with multiple draft heads, where each head predicts a different token from the same hidden representation. This intra-block multi-target supervision requires a single representation to jointly support all correlated prediction targets, encouraging the model to capture richer structures among tokens within each block.

†Correspondence to Bo Han (bhanml@comp.hkbu.edu.hk).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

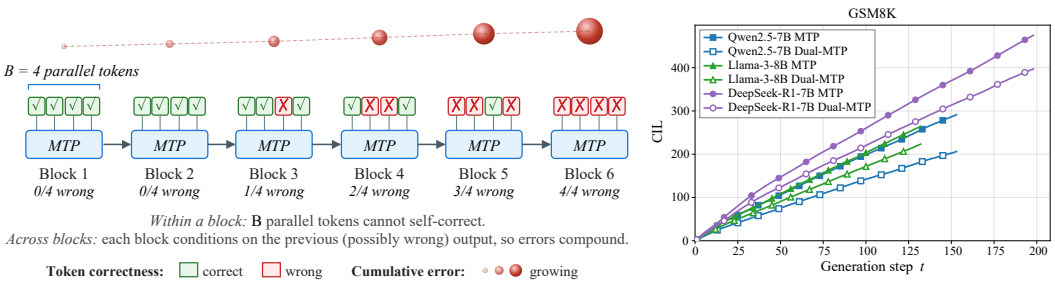


Figure 1: **Cumulative error in block-wise multi-token prediction. Left:** Example of snowball error, where early block-level mistakes propagate and amplify across subsequent predictions. **Right:** Visualization of the empirical cumulative predictive-entropy proxy across different models.

Through this mechanism, MTP learns more informative representations for multi-step prediction and has demonstrated strong empirical performance at scale.

In this paper, we revisit MTP through an information-theoretic lens. Under teacher forcing, each block-level representation is optimized with access to a clean ground-truth prefix. At inference time, however, this prefix is replaced by the model's predictions. Parallel offset heads propose tokens from a shared representation, and accepted tokens become the context for subsequently generated tokens. Consequently, even a small error in an early block can perturb the conditioning distribution that the representation was trained to expect. If predictions are sensitive to errors in the generated prefix, these perturbations can propagate through subsequently generated tokens and accumulate across successive blocks, producing a *snowball error* that degrades generation quality [10, 30] (see Figure 1). This motivates training the prediction heads to remain effective under incomplete or imperfect historical context [45].

We propose **Dual-MTP**, which jointly supervises the same targets under full and masked historical contexts using shared parameters. This objective combines learning from complete prefixes with supervision under context perturbations. We use the normalized dual loss to bound the expected Jensen–Shannon divergence between the predictions made by the same head under full and masked contexts. Across five backbones and six benchmarks, Dual-MTP achieves the highest average task score on four backbones, with gains of up to 2.46 points over MTP.

In summary, we make the following contributions:

- We express the CIL gap as conditional dependence among reference-target blocks when generation uses model outputs without reference answers. We further derive an upper bound on the divergence between full- and masked-context predictive distributions, linking the dual training objective to prediction consistency (Sections 4 and 5).
- We propose Dual-MTP, which trains shared heads to predict the same targets under full and masked contexts. In this way, it combines complete-prefix supervision with robustness to context perturbations without adding inference-time parameters (Section 5).
- We evaluate Dual-MTP across models and benchmarks, showing average-score gains on four of five backbones. Together, these experiments characterize the benefits and limitations of dual-context supervision across tasks and training configurations (Section 6).

2 Related Work

2.1 Multi-token Prediction

MTP enables a single model to predict multiple future tokens within one decoding step while remaining compatible with autoregressive verification. Prior work has explored two main paradigms to improve the performance of MTP: post-training optimization and native pre-training. Post-training methods augment pretrained LLMs with auxiliary draft heads and fine-tune only these components to extract latent future-token information [3, 22], building on early blockwise parallel decoding [29]. Recent extensions adapt this paradigm for speculative decoding, including PaSS for parallel multi-token drafting [25], and leveraging non-adjacent context for longer-horizon prediction [23]. Complementary to these, native pre-training approaches incorporate multi-token objectives directly

into training, as in ProphetNet [26], Meta MTP [12], DeepSeek-V3 [22], and MiMo [34]. Compared to generic speculative decoding, MTP offers structured parallelism without external draft models. Dual-MTP follows the post-training paradigm: it adapts a pretrained backbone with LoRA and jointly trains the added prediction heads. However, existing pre-training approaches for MTP are still grounded in local conditional likelihood objectives, focusing on improving per-step or intra-block prediction accuracy. Such objectives do not explicitly capture how errors propagate across blocks, leaving the global dependency structure under-constrained. These observations motivate training for robustness to imperfect historical context. Our dual-context objective supervises the prediction heads under both full and perturbed prefixes.

2.2 Parallel Decoding

The sequential nature of NTP has motivated extensive research on parallel decoding to improve inference efficiency. Existing approaches typically aim to amortize decoding cost by generating or verifying multiple tokens per step, often following a draft-and-verify paradigm [3, 4, 6, 11, 13]. Representative methods include speculative sampling [17, 4, 19, 18, 20], tree-based verification [33, 35, 5, 1], which differ in how draft tokens are produced and validated, and pipelined-based approaches [38, 2, 25, 39]. While these techniques can yield notable speedups under favorable conditions, they often require carefully calibrated auxiliary models, and the associated verification overhead can limit their robustness. As a result, practical efficiency gains remain highly sensitive to model behavior and task characteristics [44, 46, 47].

3 Preliminaries

Next-Token Prediction (NTP). An autoregressive language model factorizes the joint distribution over an output sequence $\mathbf{x} = (x_1, \ldots, x_L)$ as

$$p_\theta(\mathbf{x}) = \prod_{l=1}^{L} p_\theta(x_l \mid x_{<l}). \tag{1}$$

Here $x_{<l} = (x_1, \ldots, x_{l-1})$ denotes the preceding tokens available within the context window. Training uses reference prefixes; inference uses the model's own preceding outputs. The training objective is maximum likelihood estimation, which minimizes the negative log-likelihood

$$\mathcal{L}_{\text{NTP}}(\theta) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \sum_{l=1}^{L} \log p_\theta(x_l \mid x_{<l}), \tag{2}$$

where $\mathcal{D}$ denotes the training corpus. Since each position is supervised by a single target token, NTP provides one-token training signal per step and decodes tokens sequentially at inference.

Multi-Token Prediction (MTP). MTP extends NTP by partitioning the target sequence into non-overlapping blocks of B consecutive tokens and jointly predicting each block from its prefix, providing denser supervision per position. The training objective is the block-wise negative log-likelihood

$$\mathcal{L}_{\text{MTP}}(\theta) = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \sum_{s} \log p_\theta(x_{l_s:l_s+B-1} \mid x_{<l_s}), \tag{3}$$

which is typically realized by D parallel prediction heads sharing a backbone. At inference time, this design enables parallel decoding and reduces the number of sequential steps by up to a factor of B, but tokens within a block are not conditioned on verified realizations of earlier tokens in the same block. Teacher forcing supplies reference prefixes during training. At inference, parallel heads do not consume one another's proposed tokens within a block, but committed tokens form the context for later blocks.

Reference Answers and Model Outputs. Let X denote a test input and $T = (t_1, \ldots, t_N)$ its reference answer, sampled together from p_{data}. With the trained parameters θ fixed, the model generates $R_N = (r_1, \ldots, r_N)$ from X and its own preceding outputs, without access to T. Any decoding randomness is independent of T given X. Thus, T is supplied by the data, whereas R_N is produced by the model. For theoretical analysis, we represent both sequences using fixed blocks of token positions, with a padding convention for positions beyond the end of each sequence. The

verifier may accept a different number of tokens at each decoding step, so these fixed blocks need not match the accepted token groups.

Cumulative Information Loss. Following [10], the per-step information loss at step l is defined as $\mathrm{InfoLoss}(r_l) = H(t_l) - I(t_l; r_l) = H(t_l \mid r_l)$, where t_l and r_l denote the target and predicted token random variables, and r_l is conditioned on the partial response $R_{l-1} = (r_1, \ldots, r_{l-1})$. This quantity measures residual target uncertainty given the predicted token. Then the Cumulative Information Loss (CIL) up to the l-th step is defined as

$$H_{<l}(T \mid R) = \sum_{i=1}^{l-1} H(t_i \mid r_i), \tag{4}$$

where $H(t_i \mid r_i)$ denotes the entropy of the target token t_i given the predicted token r_i at the i-th step. CIL quantifies the accumulated uncertainty over multiple steps during the generation. A larger CIL indicates greater accumulated uncertainty along the generation trajectory.

4 Information-Theoretic Analysis of MTP

We relate the block-sequence CIL gap to conditional dependence among reference-target blocks when generation uses the model's own outputs without reference answers.

4.1 Block Decomposition and the Information Gap

Following the CIL in Equation (4), we analyze MTP analogously in the information-theoretical perspective. We first define the CIL within each block. For MTP, where the model predicts multiple tokens with block size B, the CIL within the block is defined below.

Definition 1 (CIL within Blocks)**.** *The block-level information loss for a block of size B at the l-th step is defined as the information loss over the entire block:*

$$\mathrm{InfoLoss}(R_l^B) = H(T_l^B) - I(T_l^B; R_l^B) = H(T_l^B \mid R_l^B) \tag{5}$$

where T_l^B denotes the block of reference tokens starting at position l, and R_l^B denotes the corresponding block of tokens predicted by the model.

Definition 1 measures the uncertainty in the reference tokens given the tokens predicted by the model for that block. For the same target block $T_l^B = (t_l, \ldots, t_{l+B-1})$ and predicted-token block $R_l^B = (r_l, \ldots, r_{l+B-1})$, the chain rule gives

$$H(T_l^B \mid R_l^B) = \sum_{i=l}^{l+B-1} H(t_i \mid t_{l:i-1}, R_l^B) \leq \sum_{i=l}^{l+B-1} H(t_i \mid r_i). \tag{6}$$

Each r_i is contained in R_l^B. Thus, for the same target block and predicted-token block, the sum of token-wise conditional entropies upper-bounds block-level CIL.

However, such block-wise modeling also introduces a structural discrepancy, as the sequence is decomposed into multiple blocks. While the information loss within each block characterizes the local prediction quality, it does not account for the potential loss induced by this decomposition. To quantify this effect, we define Δ as the difference between the sum of per-block information loss and the information loss when treating the entire sequence as a single block. Suppose a sequence is partitioned into M non-overlapping blocks. We propose the following definition.

Definition 2 (Block-Sequence CIL Gap)**.** *Let X denote the input query, shared across all blocks. Writing $\mathrm{InfoLoss}(R \mid X) := H(T \mid R, X)$ for the input-conditional version of the loss in Definition 1, the block-sequence CIL gap for a sequence of M blocks is defined as:*

$$\Delta := \sum_{m=1}^{M} \mathrm{InfoLoss}(R_m^B \mid X) - \mathrm{InfoLoss}(R_N \mid X). \tag{7}$$

The second term on the right-hand side follows from the chain rule of conditional entropy, which is

$$H(T_N \mid R_N, X) = \sum_{m=1}^{M} H(T_m^B \mid T_{<m}^B, R_N, X), \tag{8}$$

where $T_N = (T_1^B, \ldots, T_M^B)$ and $R_N = (R_1^B, \ldots, R_M^B)$. Thus,

$$\Delta = \sum_{m=1}^{M} I(T_m^B; T_{<m}^B, R_{\neq m}^B \mid R_m^B, X). \tag{9}$$

Remark 1. For each reference block T_m^B, compare conditioning on input X and its prediction R_m^B with also including preceding references $T_{<m}^B$ and other predictions $R_{\neq m}^B$. The gap sums these uncertainty reductions.

4.2 Main Result

Theorem 1 (Reference-target dependence)**.** *Let $(X, T) \sim p_{\text{data}}$ and fix the trained parameters θ and the block partition. When generation uses only X and the model's own outputs, with decoding randomness independent of T given X, $R_N \perp T \mid X$, and*

$$\Delta = \sum_{m=1}^{M} I(T_m^B; T_{<m}^B \mid X). \tag{10}$$

Proof sketch. Both the draft heads and the verifier generate from X and the preceding model outputs. Given X, the predictions are independent of the reference answer, so the two terms in Δ simplify to $\sum_m H(T_m^B \mid X)$ and $H(T \mid X)$. The chain rule expands the second term one block at a time. Subtracting gives the information that earlier reference blocks provide about each block given X. The full proof is in Appendix A.

Remark 2. For a fixed reference distribution and block partition, Δ is determined by the conditional dependence among reference-target blocks and is invariant to θ. In Section 5.1, we use the training loss to bound the divergence between the predictions made under full and masked contexts.

5 Dual-Perspective MTP

Dual-MTP trains shared prediction heads to predict the same targets from full and masked historical contexts. The dual objective combines complete-prefix supervision with context perturbations, encouraging the heads to remain predictive when parts of the historical context are unavailable.

Dual-MTP keeps the MTP architecture, a shared autoregressive backbone with parallel multiple draft heads, and operates during training. Different from the MTP, each head in Dual-MTP is supervised under both the full historical context and a stochastically masked version of it, encouraging block-level representations that encode compact, non-redundant information rather than relying on full prefix access. An overview of our method and the comparisons are shown in Figure 2.

5.1 Dual-Perspective Training

Token Stochastic Masking. To realize the masked-context branch, we apply *token stochastic masking* during training: at each step, a single binary mask is sampled over token positions with rate γ and shared across all draft heads, blocking attention to the masked positions while leaving the standard autoregressive structure intact at inference. The masking rate γ follows an annealing schedule with temperature-controlled perturbations to encourage early-stage exploration. Full details of the schedule and the attention-level masking are provided in Appendix C.

Dual-Perspective Training Objective. To stabilize training under stochastic masking, we combine supervision from both full-context and masked-context views [41]. The *full-context loss* $\mathcal{L}_{\text{full}} = -\sum \log p(\mathbf{x}_{t:t+B-1} \mid \mathbf{x}_{<t})$ uses the complete historical context $\mathbf{x}_{<t}$, whereas the *masked-context loss* $\mathcal{L}_{\text{masked}} = -\sum \log p(\mathbf{x}_{t:t+B-1} \mid \tilde{\mathbf{x}}_{<t})$ replaces $\mathbf{x}_{<t}$ with its stochastically masked version $\tilde{\mathbf{x}}_{<t}$. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{full}} + \lambda(s)\, \mathcal{L}_{\text{masked}}, \tag{11}$$

where $\lambda(s)$ is a warm-up coefficient. The two losses play complementary roles: $\mathcal{L}_{\text{full}}$ enforces *sufficiency*, ensuring block-level representations remain predictive given the complete history, while $\mathcal{L}_{\text{masked}}$ acts as a regularizer that discourages over-reliance on specific past tokens. Both losses update the shared backbone and prediction heads. The full-context branch retains supervision on the

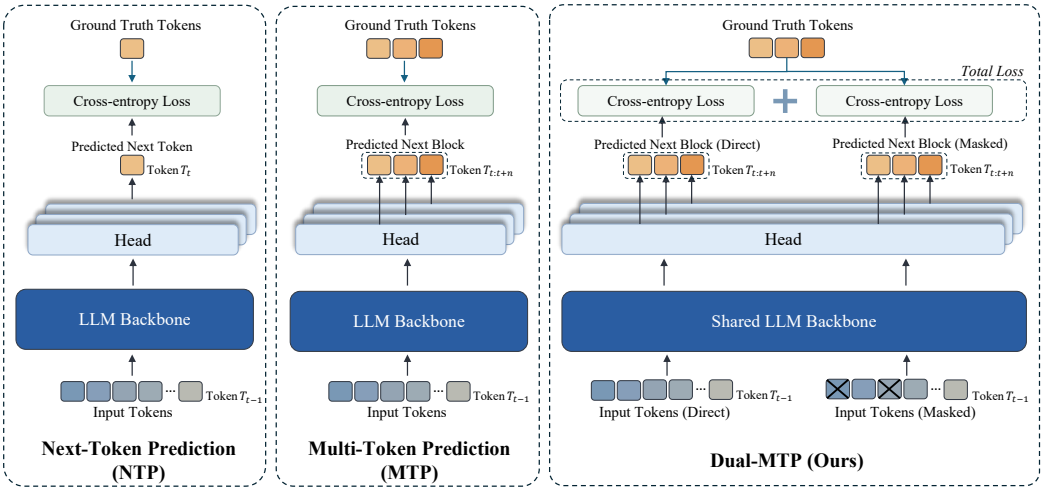


Figure 2: **Overview of NTP, MTP, and Dual-MTP.** *Left:* NTP predicts one next token. *Middle:* MTP predicts a token block via multiple heads on a shared backbone. *Right:* Dual-MTP uses full and masked training views that share the backbone and prediction heads.

uncorrupted prefix as the masked branch learns from incomplete histories. The masked-only ablation in Section 6.3 illustrates the importance of this anchor for task accuracy. Normalizing both losses over the same supervised positions allows us to analyze prediction consistency across context views.

Dual Loss and Predictive Consistency. For each prediction head and each token position it supervises, let q_f and q_m denote its predictive distributions for the same target token Y under full and masked contexts, respectively. Here $\overline{\mathcal{L}}_{\text{full}}$ and $\overline{\mathcal{L}}_{\text{masked}}$ are normalized averages of the two losses over the same data, masks, heads, and positions, with natural logarithms.

Theorem 2 (Dual loss bounds Jensen–Shannon divergence)**.** *At any training step s with $\lambda(s) > 0$, define the normalized weighted loss*

$$L_{\lambda(s)} = \overline{\mathcal{L}}_{\text{full}} + \lambda(s)\,\overline{\mathcal{L}}_{\text{masked}}.$$

Then the expected Jensen–Shannon divergence between the two predictive distributions satisfies

$$\mathbb{E}[\mathrm{JS}(q_f, q_m)] \le (\ln 2)\left[1 - \exp\left(-\frac{L_{\lambda(s)}}{\min\{1, \lambda(s)\}}\right)\right]. \tag{12}$$

Proof sketch. For a target token y, the overlap between the two distributions bounds their JS divergence by $(\ln 2)(1 - q_f(y)q_m(y))$. The product equals $e^{-\ell}$, where ℓ is the sum of the two token losses. Averaging and applying Jensen's inequality gives the stated bound because $\mathbb{E}[\ell] \le L_{\lambda(s)}/\min\{1, \lambda(s)\}$. The full proof is in Appendix B.

Remark 3. At $\lambda(s) = 1$, the bound depends on the unweighted sum of the two losses. As $L_{\lambda(s)}/\min\{1, \lambda(s)\} \to 0$, the expected divergence between the two predictions vanishes. Full-context supervision anchors prediction on the complete prefix, while masked-context supervision trains the same heads to predict from partial history.

5.2 Inference

Dual-MTP uses draft-and-verify inference [29]. The primary head and D parallel draft heads propose an ordered block of $D + 1$ tokens. The verifier checks the candidates conditioned on their preceding prefix and commits the longest accepted prefix. The verification parameter K_v specifies the top-K_v candidate set. At $K_v = 1$, multiple consecutive draft tokens can match the verifier's top-1 choices and be accepted in one iteration, reducing latency. With greedy primary-token selection, greedy fallback, and consistent tie-breaking, exact verification preserves the fixed verifier's greedy output. For $K_v > 1$, the verifier can accept additional candidates within its top-K_v set. The accepted tokens influence subsequent prefixes, making K_v a control on the accuracy–speed trade-off. We evaluate this trade-off in Appendix H.

Table 1: Main results on task performance. All results are reported as mean (%) ± std. Avg. is the arithmetic mean over six tasks; SD is over repeated decoding runs.

Model	Setting	Math500	GSM8K	CoLA	MRPC	RTE	WNLI	**Avg.** ↑
Llama-3.2-1B -Instruct	Base	13.47±1.36	30.78±2.29	78.88±1.65	77.21±3.19	66.43±4.72	55.40±5.86	53.69
	MTP	**14.40±0.35**	31.51±1.75	**79.51±1.69**	79.98±2.06	68.59±4.38	53.52±8.80	54.59
	Medusa	12.27±1.67	29.52±0.90	79.39±0.63	77.94±0.65	**72.44±0.91**	**56.81±6.95**	54.73
	Dual-MTP	14.07±2.54	**32.35±1.26**	79.23±1.27	**80.56±2.87**	71.60±1.50	53.05±9.06	**55.14**
Qwen2.5-1.5B -Instruct	Base	20.16±0.87	43.54±1.30	81.66±0.58	82.35±1.85	79.06±1.30	55.87±4.95	60.44
	MTP	**20.80±1.64**	47.64±1.54	82.42±0.29	83.33±0.98	**79.78±1.30**	55.40±5.33	61.56
	Medusa	16.87±1.63	42.99±0.57	82.04±0.49	**83.58±0.98**	78.82±2.12	**57.28±2.15**	60.26
	Dual-MTP	20.40±0.87	**48.72±1.38**	**82.71±0.34**	83.42±0.79	**79.78±1.30**	55.40±6.66	**61.74**
DeepSeek -R1-Distill -Qwen-7B	Base	26.87±0.61	64.29±1.20	60.56±0.57	38.48±0.20	53.79±2.06	56.81±1.33	50.13
	MTP	**39.27±0.66**	68.89±1.03	54.84±1.15	52.70±0.53	57.76±2.36	55.87±1.33	54.89
	Medusa	35.13±1.45	65.07±0.19	59.99±0.95	**53.02±1.55**	46.69±2.47	54.93±1.15	52.47
	Dual-MTP	37.93±0.90	**69.88±0.74**	**70.34±0.46**	50.25±1.06	**59.33±3.12**	**56.34±1.15**	**57.34**
Qwen2.5-7B -Instruct	Base	26.40±0.35	53.53±0.26	77.47±5.49	83.01±0.92	88.57±1.19	74.65±1.99	67.27
	MTP	37.53±1.37	**72.25±3.43**	77.34±5.24	84.64±1.17	89.29±0.34	77.93±3.51	73.17
	Medusa	34.20±0.72	69.02±0.93	**86.03±0.30**	**86.11±0.42**	89.65±0.95	75.59±2.66	73.43
	Dual-MTP	**38.67±0.90**	71.24±1.18	85.33±0.23	85.21±1.01	**89.77±0.61**	**80.28±1.99**	**75.08**
Llama-3-8B -Instruct	Base	13.33±0.75	44.00±2.18	85.30±0.61	88.67±0.59	88.09±1.35	71.83±1.99	65.20
	MTP	**20.07±0.09**	**56.46±0.58**	85.14±0.96	88.14±0.54	88.93±1.36	**73.71±0.66**	**68.74**
	Medusa	17.93±0.12	54.31±0.04	84.69±0.20	86.60±0.81	**89.65±0.74**	64.79±3.45	66.33
	Dual-MTP	19.27±0.41	56.10±0.86	**85.47±0.81**	**88.92±1.02**	89.41±0.90	72.30±2.39	68.58

6 Experiments

6.1 Experimental Setup

Tasks and Models. We evaluate Dual-MTP on a benchmark suite spanning two task families: (i) mathematical reasoning, with MATH500 [21] and GSM8K [7]; and (ii) natural-language understanding from GLUE [31], with CoLA, MRPC, RTE, and WNLI. This selection covers both short-form classification and reasoning-heavy generation, allowing us to test whether dual-context training transfers across these generation regimes. Experiments are conducted on five backbones: Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, Llama-3.2-1B-Instruct, Llama-3-8B-Instruct, and DeepSeek-R1-Distill-Qwen-7B.

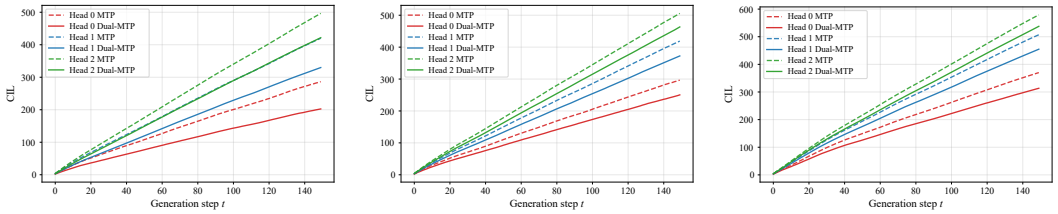
Baselines and Metrics. We compare Dual-MTP against three baselines: (i) **Base**, a supervised fine-tuning (SFT) reference in which both the backbone and the original output head are jointly trained with the standard NTP objective on the same training data and decoded autoregressively, without any draft heads; (ii) **MTP**, the post-training blockwise draft-head variant of multi-token prediction [29, 3], which shares the same draft-head architecture as Dual-MTP but is trained without stochastic masking; and (iii) **Medusa** [3], a multi-head speculative decoding method that augments the backbone with auxiliary draft heads trained jointly with the original output head. The three MTP methods use the same pretrained backbone family and head count, with method-specific loss weights and optimization schedules. The verification-parameter experiments use the configurations described in Appendix H. We report Avg@1 for math reasoning, accuracy for GLUE, except Llama-3-8B Base/MTP/Dual-MTP on MRPC, which use the mean of accuracy and F1. Math uses final-answer mathematical equivalence (Appendix F).

Implementation. We attach $D{=}3$ draft heads to the backbone, so that the backbone's primary token together with the D draft tokens form a block of size $B{=}D{+}1{=}4$, and fine-tune the backbone via LoRA jointly with the newly added heads. At inference, stochastic masking is disabled, and we run the draft-and-verify procedure of Section 5.2 with $K_v{=}5$, using at most 2048 response tokens for math reasoning and 16 for GLUE, excluding the prompt.

Training Cost. The full and masked losses are combined within each training update. Across the evaluated backbones and corpora, one epoch of Dual-MTP takes $1.26\times$–$1.50\times$ the wall-clock time of MTP. Table 8 (Appendix G) reports the training costs. Appendix H evaluates the effect of K_v on mathematical reasoning and code generation.

Table 2: Decoding throughput and end-task accuracy. Acc. is final-answer accuracy (%); tok/s and speedup retain the original timing protocol. Values are mean ± SD; speedup is relative to Base (NTP). Bold marks the best result among MTP, Medusa, and Dual-MTP.

Model	Task	Base		MTP		Medusa		**Dual-MTP**		Speedup			
		Acc.	tok/s	Acc.	tok/s	Acc.	tok/s	Acc.	tok/s	Base	MTP	Medusa	**Dual-MTP**
Llama-3.2-1B	MATH500	13.47±1.36	34.5±1.0	**14.40±0.35**	48.3±1.5	12.27±1.67	**59.1±0.6**	14.07±2.54	49.3±0.6	1.0×	1.4×	**1.7**×	1.4×
	GSM8K	30.78±2.29	35.9±0.4	31.51±1.75	53.2±3.3	29.52±0.90	**60.3±1.1**	**32.35±1.26**	55.4±2.8	1.0×	1.5×	**1.7**×	1.5×
Qwen2.5-1.5B	MATH500	20.16±0.87	20.8±0.7	**20.80±1.64**	30.2±1.5	16.87±1.63	**34.8±1.9**	20.40±0.87	31.3±2.3	1.0×	1.5×	**1.7**×	1.5×
	GSM8K	43.54±1.30	20.7±0.3	47.64±1.54	33.9±1.5	42.99±0.57	**38.3±1.9**	**48.72±1.38**	33.4±1.6	1.0×	1.6×	**1.9**×	1.6×
DeepSeek-R1-	MATH500	26.87±0.61	19.8±0.2	**39.27±0.66**	31.6±1.0	35.13±1.45	**37.4±1.5**	37.93±0.90	33.1±0.5	1.0×	1.6×	**1.9**×	1.7×
Distill-Qwen-7B	GSM8K	64.29±1.20	20.0±0.8	68.89±1.03	31.8±4.1	65.07±0.19	**42.5±0.1**	**69.88±0.74**	37.6±1.6	1.0×	1.6×	**2.1**×	1.9×
Qwen2.5-7B	MATH500	26.40±0.35	20.3±0.4	37.53±1.37	33.3±1.8	34.20±0.72	**38.7±1.7**	**38.67±0.90**	33.9±1.3	1.0×	1.6×	**1.9**×	1.7×
	GSM8K	53.53±0.26	20.1±0.2	**72.25±3.43**	29.5±0.8	69.02±0.93	**42.4±1.1**	71.24±1.18	38.2±0.2	1.0×	1.5×	**2.1**×	1.9×
Llama-3-8B	MATH500	13.33±0.75	17.8±0.2	**20.07±0.09**	27.1±0.9	17.93±0.12	**33.2±2.1**	19.27±0.41	27.8±1.0	1.0×	1.5×	**1.9**×	1.6×
	GSM8K	44.00±2.18	18.4±0.2	**56.46±0.58**	25.6±0.3	54.31±0.04	**33.9±1.7**	56.10±0.86	25.3±0.4	1.0×	1.4×	**1.8**×	1.4×



(a) Qwen2.5-7B. (b) Llama-3-8B. (c) DeepSeek-R1-Distill-Qwen-7B.

Figure 3: **Cumulative predictive entropy across draft heads.** Dual-MTP exhibits lower predictive entropy than MTP across the evaluated heads and backbones.

6.2 Main Results

Main Results. Table 1 compares five backbones on two math and four GLUE tasks. Dual-MTP attains the highest six-task average on four backbones, with gains of $+0.18$ to $+2.46$ points over MTP. On Llama-3-8B, its average is 68.58 versus MTP's 68.74. It improves over Medusa on all five backbones by $+0.41$ to $+4.87$ points, and over Base by $+1.30$ to $+7.81$ points. The largest average gain over MTP occurs on DeepSeek-R1-Distill-Qwen-7B, from 54.89 to 57.34. On GSM8K, Dual-MTP improves on MTP for three backbones by 0.84–1.09 points, but trails it on Qwen2.5-7B and Llama-3-8B by 1.01 and 0.35 points. On MATH500, it improves on MTP only for Qwen2.5-7B (38.67% versus 37.53%). GLUE contributes substantially to the average gains on larger backbones. Dual-context supervision therefore has task- and backbone-dependent benefits.

Predictive Entropy across Heads. Figure 3 compares the cumulative predictive-entropy proxy $\widehat{\mathrm{CIL}}_{<l,k}$, obtained by summing each head's predictive entropy along the generated trajectory (Appendix D). The reference-target conditional distribution is unavailable along a generated trajectory, so the plotted statistic uses the model distribution. Its relation to reference-target uncertainty depends on calibration; we therefore interpret it together with per-head prediction accuracy. Predictive entropy increases with head index. Dual-MTP lowers it at each evaluated head and backbone, with Head 0 reductions of 29.3% on Qwen2.5-7B, 15.6% on Llama-3-8B, and 15.3% on DeepSeek-R1-Distill-Qwen-7B.

Per-Head Accuracy and Training Dynamics. Figures 4 and 5 complement the predictive-entropy view of Figure 3 on GSM8K. Figure 4 shows per-head training-loss curves. Dual-MTP's mean full-context training loss is slightly higher than MTP's at each head. The mean loss gap between Head 0 and Head 2 is similar for the two methods on Qwen2.5-7B and slightly smaller under Dual-MTP on Llama-3-8B and DeepSeek-R1-Distill-Qwen-7B. Figure 5 reports per-head Avg@1 accuracy. MTP degrades sharply with head index, mirroring the monotonic CIL growth, while Dual-MTP retains higher accuracy on every head, with the largest gain at Head 1 on all three backbones. For example, on Qwen2.5-7B, Dual-MTP improves token prediction accuracy by 0.63, 1.92, and 1.26 percentage points at Heads 0, 1, and 2, respectively. For matched head–backbone pairs, Dual-MTP combines lower predictive entropy with higher head-level accuracy.

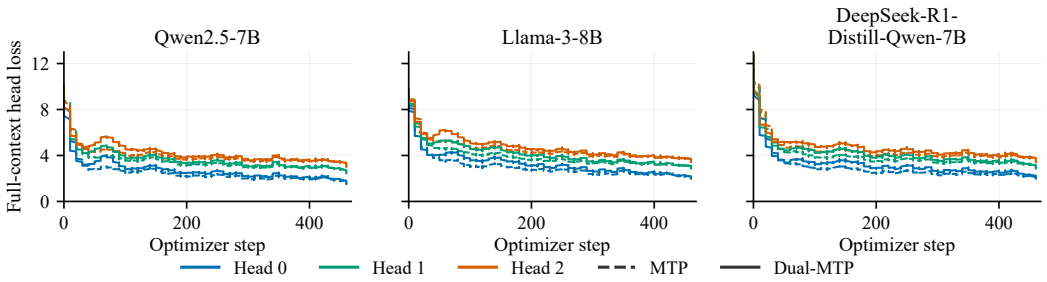


Figure 4: Per-head full-context training loss on GSM8K. Curves use a 15-observation moving average; panels share both axes. Dashed lines denote MTP and solid lines denote Dual-MTP.

Table 3: Code generation on Qwen2.5-7B-Instruct. Cells report EvalPlus pass@1 (%, mean ± sample SD) and mean speedup over fine-tuned NTP across three independent training runs.

Task	Method	$K_v = 1$	$K_v = 3$	$K_v = 5$
HumanEval+	MTP	73.37 ± 0.93 / 1.304	71.75 ± 1.53 / 1.489	71.34 ± 2.20 / 1.483
	Medusa	73.98 ± 1.27 / 1.301	72.97 ± 3.01 / 1.469	72.97 ± 3.57 / 1.467
	Dual-MTP	76.63 ± 0.35 / 1.364	74.19 ± 0.93 / 1.532	74.19 ± 0.93 / 1.530
MBPP+	MTP	70.11 ± 0.00 / 1.365	66.58 ± 1.07 / 1.503	66.23 ± 0.40 / 1.518
	Medusa	69.40 ± 0.67 / 1.366	65.61 ± 1.15 / 1.503	65.34 ± 1.40 / 1.498
	Dual-MTP	70.46 ± 0.40 / 1.344	69.22 ± 1.10 / 1.481	69.05 ± 1.73 / 1.486

Code Generation. We evaluate Qwen2.5-7B-Instruct on HumanEval+ and MBPP+ using one solution per problem. Each method trains its draft heads on greedy responses from its frozen verifier. On HumanEval+, Dual-MTP achieves 76.63% pass@1 at $K_v = 1$, compared with 73.98% for Medusa and 73.37% for MTP (Table 3). On MBPP+, MTP is the closest competitor. Its gap to Dual-MTP is only 0.35 points at $K_v = 1$ but exceeds 2.6 points at both $K_v = 3$ and 5. Thus, Dual-MTP has the highest mean pass@1 across all six comparisons, with its clearest advantage on MBPP+ under relaxed verification. Appendix H.3 gives the evaluation protocol.

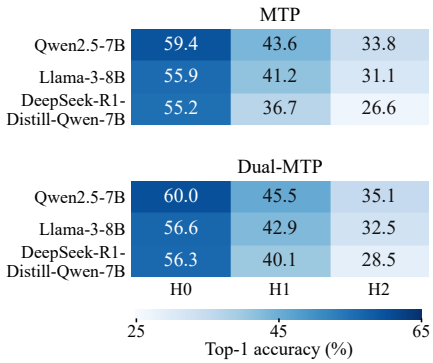


Figure 5: GSM8K top-1 accuracy by head.

Efficiency Analysis. Table 2 shows wall-clock throughput on MATH500 and GSM8K relative to NTP. Dual-MTP accelerates NTP decoding by 1.4×–1.9× across the five backbones and surpasses MTP on 8 of 10 settings, with the largest gain on Qwen2.5-7B/GSM8K (38.20 tok/s, 1.90× vs. MTP's 1.47×). Across all ten math settings, Medusa decodes faster than Dual-MTP, while Dual-MTP achieves higher accuracy. Appendix E reports per-task speculative acceptance rates across all five evaluated backbones.

Accuracy–Speed Trade-Off. Dual-MTP lies on the accuracy–speed frontier in nine of the ten settings in Figure 6. On Qwen2.5-7B/GSM8K, MTP achieves the highest accuracy (72.25%), and Medusa decodes fastest (2.11× NTP). Dual-MTP falls between them at 71.24% and 1.90× NTP. On MATH500, MTP is more accurate than Dual-MTP on four backbones, but Dual-MTP is faster on all four. MTP leads on both metrics only on GSM8K with Llama-3-8B. On HumanEval+, relaxing verification from $K_v = 1$ to 3 raises Dual-MTP's speedup from 1.364× to 1.532× but lowers pass@1 from 76.63% to 74.19% (Table 3). On MBPP+, speedup rises from 1.344× to 1.481× as pass@1 falls from 70.46% to 69.22%. At $K_v = 5$, speed changes little on either task.

6.3 Ablation Studies

Contribution of Full-Context Supervision. We evaluate Mask-only training on Qwen2.5-7B/1.5B-Instruct over GSM8K and MATH500, disabling both NTP and full-context head losses while retaining

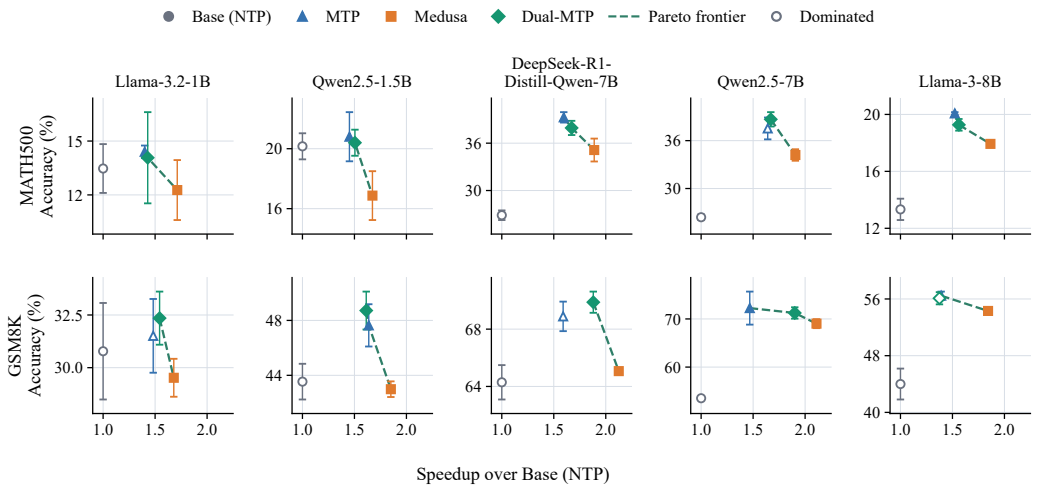


Figure 6: Math accuracy and speedup over NTP at $K_v = 5$ (Table 2). Bars show accuracy SD. Dashed lines connect nondominated means. Hollow markers denote methods matched or exceeded on both axes and surpassed on at least one.

the corresponding Dual-MTP masking settings. Mask-only accuracy is 0.00%–0.23%, compared with 20.40%–71.24% for Dual-MTP. On Qwen2.5-7B, GSM8K accuracy is 0.10% versus 71.24%, and MATH500 accuracy is 0.07% versus 38.67%. This degradation supports retaining full-context supervision when optimizing the backbone and draft heads with masked contexts. Appendix F.1 gives the configurations and reference comparisons.

Effect of Draft-Head Count. We evaluate $D \in \{1, 2, 3, 4, 5\}$ on Qwen2.5-7B-Instruct and Qwen2.5-1.5B-Instruct using all 1,319 GSM8K questions. Across ten backbone–head-count pairs, Dual-MTP exceeds MTP in accuracy in three cases, with differences from -1.82 to $+1.09$ percentage points. Acceptance rate is higher in nine cases, with a mean increase of 3.42 points and a range of -0.08 to $+15.77$ points. Higher draft acceptance therefore does not consistently imply higher task accuracy. Appendix F.2 reports the full results and the training configurations.

7 Conclusion

We introduced **Dual-MTP**, which trains shared prediction heads under full and randomly masked contexts while leaving the MTP architecture and inference procedure unchanged. Our analysis characterizes the reference-target block-sequence CIL gap and uses the normalized dual loss to bound the expected Jensen–Shannon divergence between predictions under full and masked contexts. Across five backbones and six benchmarks spanning math reasoning and GLUE understanding, Dual-MTP attains the highest average task score on four backbones, with gains of up to 2.46 points over MTP. It outperforms Medusa on average across the six tasks on every backbone, by up to 4.87 points. Original throughput is $1.4\times$–$1.9\times$ NTP. Head-level accuracy and predictive-entropy measurements provide complementary evidence of improved prediction quality, while the verification-parameter experiments illustrate the accuracy–speed trade-off.

8 Limitations

Dual-MTP has lower inference throughput than Medusa in the ten mathematical-reasoning settings of Table 2. Further work on draft generation and verification could reduce decoding latency and improve overall throughput. Our evaluation covers backbones ranging from 1B to 9B parameters. Extending the experiments to larger backbones would allow us to examine inference speed and memory requirements at larger model scales.

Acknowledgments

TYJ, ZKZ, and BH were supported by RGC General Research Fund No. 12202026, RGC Research Fellow Scheme No. RFS2627-2S02, RGC Young Collaborative Research Grant No. C2005-24Y, RGC General Research Fund No. 12200725, NSFC Major Research Plan No. 92570109, NSFC General Program No. 62376235, Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme, HKBU Faculty Niche Research Areas No. RC-FNRA-IG/25-26/SCI/05, and HKBU CSD Departmental Incentive Scheme. This work is supported in part by Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme under Grant JRCS-24-25-COMP-03, and the RGC Senior Research Fellow Scheme under Grant SRFS2324-2S02. XHJ and HDW were supported by HK RGC research grant No.: SRFS2425-1S01.

References

[1] Oscar Brown, Zhengjie Wang, Andrea Do, Nikhil Mathew, and Cheng Yu. Dynamic depth decoding: Faster speculative decoding for llms. *arXiv preprint arXiv:2409.00142*, 2024.

[2] Branden Butler, Sixing Yu, Arya Mazaheri, and Ali Jannesari. Pipeinfer: Accelerating llm inference using asynchronous pipelined speculation. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–19. IEEE, 2024.

[3] 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 *International Conference on Machine Learning*, pages 5209–5235. PMLR, 2024.

[4] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[5] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

[6] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin C Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37: 86226–86242, 2024.

[7] 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.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[9] Xiao Feng, Bo Han, Zhanke Zhou, Jiaqi Fan, Jiangchao Yao, Ka Ho Li, Dahai Yu, and Michael Kwok-Po Ng. RewardFlow: Topology-aware reward propagation on state graphs for agentic RL with large language models. *arXiv preprint arXiv:2603.18859*, 2026.

[10] Zeyu Gan, Yun Liao, and Yong Liu. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. In *Forty-second International Conference on Machine Learning*, 2025.

[11] Xinwei Geng, Xiaocheng Feng, and Bing Qin. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308, 2021.

[12] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, pages 15706–15734, 2024.

[13] Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in neural information processing systems*, 32, 2019.

[14] 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.

[15] Hangfeng He and Weijie J Su. A law of next-token prediction in large language models. *Physical Review E*, 112(3):035317, 2025.

[16] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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

[18] 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*, pages 7421–7432, 2024.

[19] 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*, pages 28935–28948. PMLR, 2024.

[20] 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.

[21] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024.

[22] 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.

[23] Xiaohao Liu, Xiaobo Xia, Weixiang Zhao, Manyi Zhang, Xianzhi Yu, Xiu Su, Shuo Yang, See-Kiong Ng, and Tat-Seng Chua. L-MTP: Leap multi-token prediction beyond adjacent context for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[24] Nguyen Nhat Minh, Andrew Baker, Clement Neo, Allen G Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023.

[26] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, 2020.

[27] Qwen Team. Qwen3.5-4B Model Card. `https://huggingface.co/Qwen/Qwen3.5-4B`, 2026.

[28] Qwen Team. Qwen3.5-9B Model Card. `https://huggingface.co/Qwen/Qwen3.5-9B`, 2026.

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

[30] Hongduan Tian, Xiao Feng, Ziyuan Zhao, Xiangyu Zhu, Rolan Yan, and Bo Han. Multi-agent debate with memory masking. In *International Conference on Learning Representations (ICLR)*, 2026.

[31] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018.

[32] Jianxun Wang and Yixiang Chen. A review on code generation with llms: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pages 284–289. IEEE, 2023.

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

[34] Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

[35] Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, and Lei Zou. Dyspec: Faster speculative decoding with dynamic token tree structure. *World Wide Web*, 28(3):36, 2025.

[36] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL `https://arxiv.org/abs/2407.10671`.

[37] 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.

[38] Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact LLM decoding. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

[39] Hanling Yi, Feng Lin, Hongbin Li, Ning Peiyang, Xiaotian Yu, and Rong Xiao. Generation meets verification: Accelerating large language model inference with smart parallel auto-correct decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5285–5299, 2024.

[40] Zihao Yi, Jiarui Ouyang, Zhe Xu, Yuwen Liu, Tianhao Liao, Haohao Luo, and Ying Shen. A survey on recent advances in llm-based multi-turn dialogue systems. *ACM Computing Surveys*, 58(6):1–38, 2025.

[41] Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. In *International Conference on Learning Representations*, 2026.

[42] Zoie Zhao, Sophie Song, Bridget Duah, Jamie Macbeth, Scott Carter, Monica P Van, Nayeli Suseth Bravo, Matthew Klenk, Kate Sick, and Alexandre LS Filipowicz. More human than human: Llm-generated narratives outperform human-llm interleaved narratives. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 368–370, 2023.

[43] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*, 2023.

[44] Zhanke Zhou, Chentao Cao, Xiao Feng, Xuan Li, Zongze Li, Xiangyu Lu, Jiangchao Yao, Weikai Huang, Tian Cheng, Jianghangfan Zhang, Tangyu Jiang, Linrui Xu, Yiming Zheng, Brando Miranda, Tongliang Liu, Sanmi Koyejo, Masashi Sugiyama, and Bo Han. AlphaApollo: A system for deep agentic reasoning. *arXiv preprint arXiv:2510.06261*, 2025.

[45] Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=LCaTpVuvpj`.

[46] Zhanke Zhou, Zongze Li, Weikai Huang, Xuan Li, Chentao Cao, Xiao Feng, Xiangyu Lu, Jinbo Hu, Menghan Lu, Yi Xie, Nico Pelleriti, Shiyang Liu, Max Zimmer, Brando Miranda, Jiangchao Yao, Bo Liu, Sanmi Koyejo, Sebastian Pokutta, and Bo Han. AlphaDiana: A measurement system for agents in verifiable reasoning tasks. *arXiv preprint arXiv:2610.20675*, 2026.

[47] Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for LLM reasoning. In *ICML*, 2026.

[48] Zhanke Zhou, Zhaocheng Zhu, Xuan Li, Mikhail Galkin, Xiao Feng, Sanmi Koyejo, Jian Tang, and Bo Han. Landscape of thoughts: Visualizing the reasoning process of large language models. In *ICLR*, 2026.

A Proof of Theorem 1

A.1 Reference Targets and Generated Predictions

The training procedure produces parameters θ, which are held fixed throughout the inference analysis. A test input and reference target are drawn from $p_{\mathrm{data}}(X, T)$. We fix a finite horizon and a block partition; end-of-sequence padding handles shorter outputs. Write $T = (T_1^B, \ldots, T_M^B)$ and $R_N = (R_1^B, \ldots, R_M^B)$. Each block in the analysis consists of a fixed group of output positions.

A.2 Decoding Assumption

Let π_θ denote the conditional distribution over the next output block induced by the decoding procedure. The decoder has access only to X, its preceding predictions, and randomness independent of the reference target given X. Thus

$$p(R_N \mid X, T) = \prod_{m=1}^{M} \pi_\theta(R_m^B \mid X, R_{<m}^B) = p(R_N \mid X), \tag{13}$$

and hence

$$R_N \perp T \mid X. \tag{14}$$

Predictions and targets share the input X. Conditional on X, the decoder generates R_N using its own predictions and independent randomness, which gives Equation (14).

A.3 Proof

Proof. Conditional independence implies

$$H(T_m^B \mid R_m^B, X) = H(T_m^B \mid X), \qquad H(T \mid R_N, X) = H(T \mid X). \tag{15}$$

Substitution into the definition of Δ and the conditional entropy chain rule give

$$\begin{aligned} \Delta &= \sum_{m=1}^{M} H(T_m^B \mid X) - H(T \mid X) \\ &= \sum_{m=1}^{M} \big[H(T_m^B \mid X) - H(T_m^B \mid T_{<m}^B, X)\big] \\ &= \sum_{m=1}^{M} I(T_m^B; T_{<m}^B \mid X). \end{aligned} \tag{16}$$

□

Remark. For a fixed reference distribution and partition, the input-conditional gap is invariant to the trained parameters. It equals zero when the reference target is deterministic given X. Thus, the identity characterizes the dependence structure of the reference targets; the loss-dependent analysis below characterizes predictive consistency across context views.

B Proof of Theorem 2

Definitions and Normalization. For the same training example, mask, head, and supervised position, let Y be the target token and q_f, q_m the predictive distributions under full and masked contexts. With $q_a = (q_f + q_m)/2$ and natural logarithms, their Jensen–Shannon divergence is

$$\mathrm{JS}(q_f, q_m) = \tfrac{1}{2}\,\mathrm{KL}(q_f \| q_a) + \tfrac{1}{2}\,\mathrm{KL}(q_m \| q_a). \tag{17}$$

We use the same normalized weights over examples, masks, heads, and positions in all expectations. The sum of the two mean token losses is

$$J = \mathbb{E}[-\ln q_f(Y) - \ln q_m(Y)] = \overline{\mathcal{L}}_{\mathrm{full}} + \overline{\mathcal{L}}_{\mathrm{masked}}. \tag{18}$$

A Pointwise Bound. For two distributions p, q on the token vocabulary, set $a_z = (p_z + q_z)/2$ and $d_z = |p_z - q_z|/(p_z + q_z)$ when $a_z > 0$. Their contribution to Jensen–Shannon divergence is $a_z f(d_z)$, where

$$f(d) = \tfrac{1}{2}\big[(1+d)\ln(1+d) + (1-d)\ln(1-d)\big].$$

Since f is convex on $[0,1]$, $f(0) = 0$, and $f(1) = \ln 2$, it lies below the chord $d \ln 2$. Consequently,

$$\mathrm{JS}(p,q) \le (\ln 2) \sum_z a_z d_z = (\ln 2)\,\mathrm{TV}(p,q).$$

Also $\mathrm{TV}(p,q) = 1 - \sum_z \min\{p_z, q_z\} \le 1 - \min\{p_y, q_y\}$ for any target token y. Since $p_y q_y \le \min\{p_y, q_y\}$, setting $\ell = -\ln q_f(y) - \ln q_m(y)$ gives

$$\begin{aligned}\mathrm{JS}(q_f, q_m) &\le (\ln 2)\,\mathrm{TV}(q_f, q_m) \\ &\le (\ln 2)\big(1 - \min\{q_f(y), q_m(y)\}\big) \\ &\le (\ln 2)\big(1 - q_f(y) q_m(y)\big) = (\ln 2)(1 - e^{-\ell}). \end{aligned} \tag{19}$$

Zero target probability is handled by the limit $\ell = +\infty$.

Averaging and the Weighted Objective. The function $u \mapsto 1 - e^{-u}$ is concave. Averaging the pointwise inequality over the same examples, masks, heads, and token positions and applying Jensen's inequality proves

$$\mathbb{E}[\mathrm{JS}(q_f, q_m)] \le (\ln 2)\mathbb{E}[1 - e^{-\ell}] \le (\ln 2)(1 - e^{-\mathbb{E}\ell}) = (\ln 2)(1 - e^{-J}).$$

The upper envelope has derivative $(\ln 2)e^{-J} > 0$. Hence, reducing J tightens the bound, and $J \to 0$ implies $\mathbb{E}[\mathrm{JS}(q_f, q_m)] \to 0$.

Fix any training step s with $\lambda(s) > 0$. Nonnegative cross-entropies give

$$L_{\lambda(s)} = \overline{\mathcal{L}}_{\text{full}} + \lambda(s)\,\overline{\mathcal{L}}_{\text{masked}} \ge \min\{1, \lambda(s)\} J.$$

Consequently, $J \le L_{\lambda(s)} / \min\{1, \lambda(s)\}$. Monotonicity of the preceding upper envelope yields

$$\mathbb{E}[\mathrm{JS}(q_f, q_m)] \le (\ln 2)(1 - e^{-J}) \le (\ln 2)\left[1 - \exp\left(-\frac{L_{\lambda(s)}}{\min\{1, \lambda(s)\}}\right)\right],$$

which proves Theorem 2. The inequality holds pointwise in s, allowing $\lambda(s)$ to vary over the warm-up schedule. For $\lambda(s) = 1$, $L_{\lambda(s)} = J$ and the unweighted bound is recovered.

Prediction consistency follows when $L_{\lambda(s)} / \min\{1, \lambda(s)\} \to 0$. In particular, if $\lambda(s)$ is uniformly bounded below by a positive constant, $L_{\lambda(s)} \to 0$ suffices. At a step with $\lambda(s) = 0$, the weighted objective contains only full-context supervision and the weighted bound does not apply; the unweighted bound in terms of J remains valid.

C Token Stochastic Masking: Implementation Details

This appendix provides the full specification of the token stochastic masking mechanism described in Section 5.1. At each training step, a single binary mask over token positions is sampled and shared across all prediction heads. This design constrains access to historical context during training while preserving standard autoregressive inference behavior, since masking is disabled at inference time.

C.1 Annealed Masking Rate

We adopt an annealing schedule to dynamically adjust the masking rate throughout training. Let s denote the current training step and S the maximum number of steps. We define the training progress as $p = \min(s/S, 1)$. The base masking rate is computed via exponential decay:

$$\tilde{\gamma}(p) = \gamma_{\text{final}} + (\gamma_{\text{init}} - \gamma_{\text{final}}) \exp(-\alpha p), \tag{20}$$

where γ_{init} and γ_{final} are the initial and final mask rates and α controls the decay.

To encourage exploration early in training, we further introduce temperature-controlled stochastic perturbations:

$$\gamma = \tilde{\gamma}(p) + \epsilon, \qquad \epsilon \sim \mathcal{N}\big(0, \sigma^2(p)\big), \tag{21}$$

where the variance $\sigma^2(p)$ is proportional to a temperature parameter that decays over training. The resulting masking rate is clipped to a predefined range to ensure stability.

C.2 Mask Construction and Application

Given a masking rate γ, we sample a binary mask over token positions as

$$M_i \sim \text{Bernoulli}(1-\gamma). \tag{22}$$

The sampled mask is incorporated into the attention computation. Concretely, for the attention logits $A^{(h)} \in \mathbb{R}^{Q\times K}$ corresponding to head h, we apply the transformation

$$A^{(h)}_{q,i} \leftarrow \begin{cases} A^{(h)}_{q,i}, & M_i = 1, \\ -\infty, & M_i = 0, \end{cases} \tag{23}$$

prior to the softmax operation. This prevents queries in the current block from attending to masked tokens. All attention heads use the same mask at each training step.

D Empirical Predictive-Entropy Proxy

D.1 Definition

For head k at generation position i, let h_i be the recorded hidden state and $q_{i,k} = P_{\theta_k}(\cdot \mid h_i)$ its vocabulary distribution. The plotted quantity is

$$\widehat{H}_{i,k} = H(q_{i,k}) = -\sum_{x\in\mathcal{V}} q_{i,k}(x)\ln q_{i,k}(x), \tag{24}$$

$$\widehat{\text{CIL}}_{<l,k} = \sum_{i=1}^{l-1} \widehat{H}_{i,k}. \tag{25}$$

This statistic measures cumulative uncertainty in the model distributions along a generated trajectory. It conditions on hidden states h_i, while the reference-target CIL in Equation (4) conditions on predicted tokens r_i.

D.2 Calibration-Dependent Interpretation

Let $p_i = p_{\text{data}}(T_i \in \cdot \mid h_i)$ denote the conditional reference distribution at the same inference states, and define the state-conditioned quantity $\text{CIL}^{\text{state}}_{<l} = \sum_{i<l} H(p_i)$. The following result relates the predictive-entropy statistic to this reference quantity through the discrepancy between p_i and $q_{i,k}$.

Proposition 1 (State-conditioned surrogate)**.** *For a finite vocabulary of size $V \geq 2$, let $\delta_{i,k} = \frac{1}{2}\|p_i - q_{i,k}\|_1$. If $\delta_{i,k} \leq 1 - 1/V$ at each position, then*

$$\left|\widehat{\text{CIL}}_{<l,k} - \text{CIL}^{\text{state}}_{<l}\right| \leq \sum_{i<l} \left[\delta_{i,k}\ln(V-1) + h_2(\delta_{i,k})\right], \tag{26}$$

where h_2 is binary entropy with natural logarithms. Equality of the two cumulative quantities follows if $p_i = q_{i,k}$ for every position. For a fixed horizon, the difference vanishes if the forward KL errors tend to zero at all these states.

Proof. Apply the finite-alphabet entropy continuity bound to each pair $(p_i, q_{i,k})$ and then the triangle inequality [8]. Pinsker's inequality gives $\delta_{i,k} \leq \sqrt{\text{KL}(p_i\|q_{i,k})/2}$, so convergence of the state-wise KL errors and continuity of the right-hand side prove the final statement. □

The approximation quality depends on distributional agreement along the model's generated trajectories. We therefore evaluate predictive entropy jointly with task and head-level accuracy to distinguish more accurate predictions from overconfidence.

E Speculative Acceptance Rate Analysis

Table 4 compares the fraction of draft tokens accepted by the verifier. Dual-MTP generally achieves higher or comparable acceptance rates to MTP, indicating closer agreement between its draft predictions and verifier decisions. Medusa achieves higher average acceptance rates on most backbones,

Table 4: Speculative decoding acceptance rate (mean ± SD over three decoding runs).

Model	Setting	Math500	GSM8K	CoLA	MRPC	RTE	WNLI	**Avg. ↑**
Llama-3.2-1B -Instruct	MTP	49.1±0.3	52.5±0.5	57.1±1.7	64.1±6.1	50.0±0.0	50.0±0.0	53.8
	Medusa	**56.5±0.7**	**62.0±0.6**	**59.2±0.2**	**65.6±0.7**	**50.0±0.0**	50.0±0.0	**57.2**
	Dual-MTP	49.1±0.1	52.6±0.6	56.3±1.8	64.6±6.3	50.0±0.0	50.0±0.0	53.8
Qwen2.5-1.5B -Instruct	MTP	47.7±0.3	49.8±0.4	**66.3±0.0**	61.2±0.3	44.9±0.1	**66.2±0.5**	56.0
	Medusa	**55.0±0.4**	**59.5±0.4**	54.0±0.2	**62.2±0.7**	**45.1±0.4**	65.5±0.3	**56.9**
	Dual-MTP	48.3±0.4	50.4±0.4	56.2±0.7	61.1±0.4	44.9±0.1	63.2±0.1	54.0
DeepSeek -R1-Distill-Qwen-7B	MTP	52.0±0.7	55.2±0.3	**26.5±0.3**	43.2±1.0	**12.7±0.2**	**34.1±0.8**	37.3
	Medusa	**60.0±0.6**	**65.6±0.2**	24.1±0.2	29.4±1.4	11.9±0.4	32.9±0.5	37.3
	Dual-MTP	52.3±0.5	58.5±0.4	24.2±0.6	**44.8±1.1**	11.8±1.8	32.0±0.5	**37.3**
Qwen2.5-7B -Instruct	MTP	53.2±0.3	41.2±0.7	65.7±0.5	**87.6±0.8**	**61.9±2.0**	57.1±0.5	61.1
	Medusa	**61.2±0.2**	**65.5±0.1**	**71.1±0.1**	69.4±0.2	49.3±0.0	71.4±0.2	64.7
	Dual-MTP	53.2±0.2	57.1±0.1	66.8±0.3	85.9±0.4	61.0±0.6	**72.7±0.9**	**66.1**
Llama-3-8B -Instruct	MTP	53.4±0.1	40.7±0.4	54.2±0.7	**73.3±1.9**	50.0±0.0	50.8±0.4	53.7
	Medusa	**60.4±0.3**	**67.9±0.6**	**55.0±0.0**	72.4±0.1	**50.0±0.0**	50.0±0.0	**59.3**
	Dual-MTP	55.3±0.6	62.9±0.4	54.4±0.6	71.6±3.9	50.0±0.0	**50.8±0.2**	57.5

while Dual-MTP achieves higher average task scores on all five backbones (Tables 1 and 4). The methods use the same verification rule at a given threshold; differences in acceptance arise from their learned draft and verifier distributions. Together, Tables 1, 2 and 4 characterize prediction accuracy, accepted draft length, and decoding throughput.

F Ablation Studies: Full Results

We compare training objectives, the number of draft heads, and the warm-up schedule for the masked loss. All three ablations use $K_v = 5$.

Scoring. All mathematical-reasoning accuracies use the same final-answer rule on at most 2048 response tokens, excluding the prompt. We extract the final answer from a boxed expression, an answer label, a concluding statement, a final equation, or a standalone answer. We count it as correct if it is mathematically equivalent to the reference answer. Missing, incomplete, unparseable, or contradictory final answers count as incorrect; occurrences of the reference within intermediate reasoning do not establish correctness. Historical responses are truncated before rescoring, while new evaluations enforce the limit during generation. In Tables 1, 2, and 5, SD is the sample SD for 1B/1.5B models and population SD for 7B/8B models, over repeated decoding runs. Other repeated mathematical evaluations report sample SD; SD is omitted for single runs.

GLUE scores use at most 16 response tokens, excluding the prompt, with task-specific classification parsing. Saved answers are tokenized with the corresponding model tokenizer and truncated before scoring. SD uses the same model-size convention as Table 1. We average two runs for Llama-3-8B Base/MRPC and Dual-MTP/CoLA and three for all other cells.

F.1 Ablation Study: Necessity of the Dual Training Objective

We train Mask-only with the NTP and full-context head-loss weights set to zero. Mask-only retains the masking and training settings of the corresponding Dual-MTP run. MTP and Dual-MTP reference accuracies come from Table 1. The 1.5B/MATH500 MTP reference has a different full-loss weight from Dual-MTP; the 7B/GSM8K MTP reference was trained on MATH7500, whereas Dual-MTP and Mask-only used GSM8K.

Compared with Dual-MTP, Mask-only loses 71.14 points on 7B/GSM8K, 38.60 on 7B/MATH500, 48.50 on 1.5B/GSM8K, and 20.40 on 1.5B/MATH500. The near-zero final-answer accuracy shows that masked-context supervision alone is insufficient when jointly training the backbone and heads.

Table 5: Training-objective comparison. Accuracy (%) is mean ± SD over three decoding runs; 1.5B uses sample SD and 7B uses population SD. MTP and Dual-MTP use the references in Table 1.

Model	Method	GSM8K	MATH500
Qwen2.5-7B	Mask-only	0.10±0.04	0.07±0.09
	MTP	72.25±3.43	37.53±1.37
	Dual-MTP	71.24±1.18	38.67±0.90
Qwen2.5-1.5B	Mask-only	0.23±0.08	0.00±0.00
	MTP	47.64±1.54	20.80±1.64
	Dual-MTP	48.72±1.38	20.40±0.87

F.2 Draft-Head Count Sweep

We vary the number of draft heads from one to five on both backbones. At $D = 3$, we reuse the checkpoints from the main comparison. The 7B MTP checkpoint was trained on MATH7500, while the other head-count configurations use GSM8K.

Table 6: Draft-head count sweep on all 1,319 GSM8K questions. Accuracy and acceptance rate (AR), in %, are mean ± sample SD over three decoding runs. Bold marks the more accurate method.

Backbone	D	MTP Acc.	MTP AR	Dual-MTP Acc.	Dual-MTP AR
Qwen2.5-7B	1	70.91±0.70	64.56±0.75	**71.97±2.39**	66.81±0.98
	2	**71.34±1.39**	57.72±0.69	70.26±1.48	60.04±0.67
	3	**72.25±4.20**	40.96±0.90	71.24±1.44	56.73±0.56
	4	**71.32±0.68**	53.03±0.49	70.86±1.14	55.62±0.63
	5	**72.00±1.58**	52.72±0.34	70.18±0.80	55.42±0.42
Qwen2.5-1.5B	1	**47.38±0.55**	56.55±0.93	46.63±1.05	58.52±0.93
	2	**48.29±0.67**	49.58±0.56	47.23±0.80	51.46±0.56
	3	47.64±1.54	50.00±0.52	**48.72±1.38**	49.92±0.44
	4	47.92±1.90	45.41±0.28	**48.12±0.91**	47.87±0.24
	5	**49.58±0.77**	45.46±0.22	48.09±0.90	47.80±0.24

Three observations follow from Table 6. (i) Dual-MTP improves accuracy in three of ten pairs: 1.5B at $D = 3, 4$ and 7B at $D = 1$. Differences range from -1.82 to $+1.09$ points. (ii) Acceptance rate improves in nine pairs, by 3.42 points on average; the range is -0.08 to $+15.77$ points. The largest increase is at $D = 3$ on 7B, where accuracy is nevertheless lower. (iii) The highest observed accuracy is at $D = 5$ for 1.5B MTP, $D = 3$ for 1.5B Dual-MTP and 7B MTP, and $D = 1$ for 7B Dual-MTP. The preferred head count therefore depends on both the backbone and training objective.

F.3 Warm-up Coefficient $\lambda(s)$

We compare linear warm-up of the masked-loss weight with a constant $\lambda = 1$ on the full MATH500 and GSM8K test sets.

Warm-up improves accuracy in all four settings. The largest gain is on Qwen2.5-7B/MATH500: 40.80% versus 37.40%, or 3.40 points. The 1.5B gains are 1.20 points on MATH500 and 2.27 on GSM8K. The 7B/GSM8K scores are nearly equal (69.98% and 69.90%). We adopt linear warm-up.

G Training Computation and Cost

The full and masked views use the same backbone and draft heads, and their weighted losses are added before backpropagation. Computing the second view adds work per example. When prefix reuse is enabled, the two views share computation up to the first masked token, and gradients from both losses flow through the shared prefix. A longer shared prefix therefore saves more computation.

Across five model families and two training corpora, one epoch of Dual-MTP takes $1.26\times$–$1.50\times$ as long as one epoch of MTP (Table 8).

Table 7: Linear warm-up versus a constant dual-loss coefficient $\lambda = 1.0$. Values are from one evaluation per configuration. Accuracy uses the final-answer metric on at most 2048 response tokens. Throughput and acceptance rate (AR, a fraction) retain the original generation measurements.

Backbone / Task	Setting	Acc(%)	tok/s	AR
Qwen2.5-7B / MATH500	warmup	**40.80**	31.0	0.549
	λ=1.0	37.40	34.5	0.561
Qwen2.5-7B / GSM8K	warmup	**69.98**	34.8	0.572
	λ=1.0	69.90	37.6	**0.585**
Qwen2.5-1.5B / MATH500	warmup	**18.20**	27.8	0.483
	λ=1.0	17.00	30.2	**0.496**
Qwen2.5-1.5B / GSM8K	warmup	**48.45**	34.3	0.514
	λ=1.0	46.17	33.6	**0.522**

Table 8: One-epoch GPU-hours (MTP / Dual-MTP) and wall-clock time ratios (Dual / MTP).

Model	Training corpus	Dual / MTP	GPU-hours (MTP / Dual)
Qwen2.5-1.5B-Instruct	GSM8K	1.26	0.329 / 0.414
Qwen2.5-1.5B-Instruct	MATH7500	1.37	0.342 / 0.468
Qwen2.5-7B-Instruct	GSM8K	1.30	0.344 / 0.447
Qwen2.5-7B-Instruct	MATH7500	1.50	0.375 / 0.563
DeepSeek-R1-Distill-Qwen-7B	GSM8K	1.27	0.340 / 0.434
DeepSeek-R1-Distill-Qwen-7B	MATH7500	1.41	0.377 / 0.531
Llama-3.2-1B-Instruct	GSM8K	1.30	0.200 / 0.260
Llama-3.2-1B-Instruct	MATH7500	1.35	0.219 / 0.295
Llama-3-8B-Instruct	GSM8K	1.31	0.360 / 0.473
Llama-3-8B-Instruct	MATH7500	1.41	0.436 / 0.613

H Effect of the Verification Parameter K_v

The verification parameter K_v controls which draft candidates can be accepted by the verifier. We evaluate its effect on mathematical reasoning and code generation using Qwen2.5-7B-Instruct with $D = 3$ draft heads. We first examine Dual-MTP across five values of K_v at a fixed checkpoint, then present cross-method comparisons at $K_v \in \{1, 3, 5\}$.

H.1 Fixed-Checkpoint Analysis on MATH500

We hold the Qwen2.5-7B-Instruct checkpoint fixed after training on MATH7500 with three draft heads. We vary K_v on the full MATH500 test set and score each final answer as in Section 6.

Table 9: Dual-MTP performance across verification parameters on MATH500. Results are mean $\pm$ sample SD over three decoding runs at a fixed checkpoint.

K_v	Correct / 500 by run	Correctness (%)	Throughput (tok/s)
1	345 / 330 / 341	67.73$\pm$1.55	118.45 $\pm$ 0.90
2	311 / 307 / 305	61.53$\pm$0.61	124.97 $\pm$ 0.99
3	295 / 307 / 300	60.13$\pm$1.21	125.91 $\pm$ 0.41
4	289 / 293 / 306	59.20$\pm$1.78	127.03 $\pm$ 0.47
5	292 / 292 / 302	59.07$\pm$1.15	127.63 $\pm$ 0.15

Increasing K_v from 1 to 5 raises throughput from 118.45 to 127.63 tok/s, while final-answer accuracy changes from 67.73% to 59.07%. The timing reference for this fixed-checkpoint experiment is the pretrained NTP model without LoRA, evaluated on the same GPU, dataset, scoring procedure, generation settings, and timing protocol. Its throughput is 83.9 tok/s. The corresponding speedups for $K_v = 1, 2, 3, 4, 5$ are $1.41\times$, $1.49\times$, $1.50\times$, $1.51\times$, and $1.52\times$.

H.2 Mathematical Reasoning

Table 10 presents the MTP, Medusa, and Dual-MTP results on MATH500 and GSM8K at $K_v = 1, 3, 5$. Accuracy uses the shared final-answer rule on at most 2048 response tokens. Speedup retains the original timing measurements. GSM8K Dual-MTP uses one fixed checkpoint across three decoding runs, summarized by mean and sample SD.

Table 10: Mathematical reasoning at three verification settings. Cells give accuracy (%, mean ± sample SD over three runs) and speedup over NTP.

Task	Method	$K_v = 1$	$K_v = 3$	$K_v = 5$
MATH500	MTP	47.73±2.44 / 1.46	39.13±4.01 / 1.61	37.80±3.61 / 1.61
	Medusa	44.73±3.35 / 1.80	35.33±1.53 / 1.91	35.20±0.87 / 1.92
	Dual-MTP	67.73±1.55 / 1.41	60.13±1.21 / 1.50	59.07±1.15 / 1.52
GSM8K	MTP	78.52±0.64 / 1.56	73.11±0.97 / 1.65	72.73±1.26 / 1.66
	Medusa	77.56±0.47 / 1.85	69.50±2.50 / 1.96	69.40±1.50 / 1.98
	Dual-MTP	86.45±1.66 / 1.43	82.16±1.41 / 1.63	82.18±0.65 / 1.65

H.3 Code Generation

We train each verifier on the same 266 decontaminated MBPP examples, then freeze it and use its greedy responses to train its three draft heads. At evaluation, each model generates one solution of at most 512 tokens for each of the 164 HumanEval+ and 378 MBPP+ problems. We run the same EvalPlus tests for all methods and report mean pass@1 and sample SD across three independent training runs (Table 3). To measure speedup, we time the first 50 problems against fine-tuned NTP on the same GPU and average the per-run ratios. The NTP baseline scores $73.58 \pm 1.27\%$ on HumanEval+ and $70.11 \pm 0.46\%$ on MBPP+.

Accuracy–Speed Trade-Off. Dual-MTP has the highest mean pass@1 among the three draft-head methods at every evaluated K_v on both benchmarks. On HumanEval+, it also has the highest mean speedup at each K_v and improves on the NTP reference in both metrics. Its $K_v = 1$ point attains 76.63% pass@1 at $1.364\times$ speedup; $K_v = 3$ trades some accuracy for $1.532\times$ speedup at 74.19% pass@1. On MBPP+, Dual-MTP at $K_v = 1$ improves on NTP in both metrics. At $K_v = 3/5$, its pass@1 falls below NTP, while MTP reaches higher speeds at lower accuracy. Figure 7 shows these measured operating points and their Pareto frontiers.

I Additional Experiments

I.1 Training-Duration Comparison

We compare two epochs of MTP with one epoch of Dual-MTP on Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. Both methods train on the same examples for each task, using MATH7500 for MATH500 and the GSM8K training split for GSM8K.

The two 1.5B configurations and the 7B/MATH500 configuration were selected after examining target-task evaluation results; the 7B/GSM8K configuration was fixed.

Each entry in Table 11 is a single evaluation on the full test set using the final-answer metric in Section 6. Dual-MTP obtains higher accuracy in three comparisons, by 0.53–3.00 points; on 7B/MATH500 it scores 41.00% versus MTP's 41.40%. With half as many optimizer updates, Dual-MTP solves 7 more GSM8K problems at 1.5B and 12 more at 7B. Epoch count specifies data exposure; the per-epoch wall-clock costs are reported separately in Table 8.

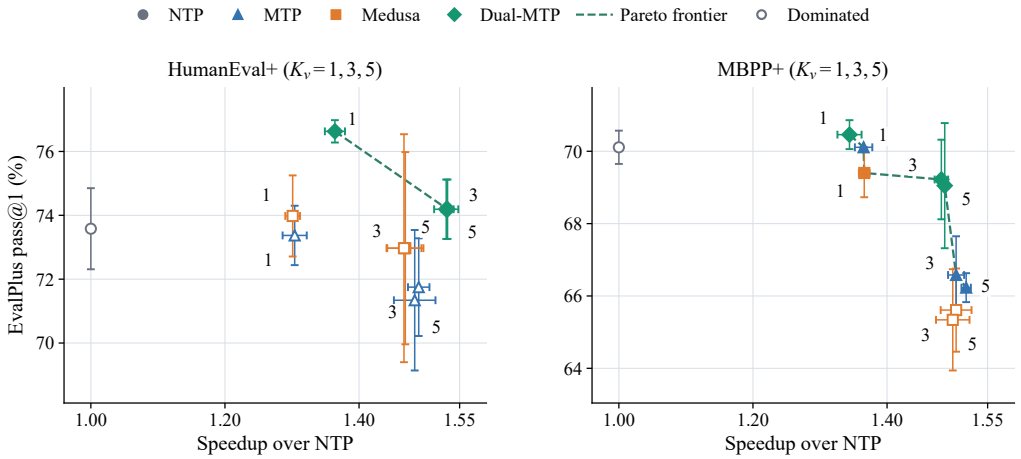


Figure 7: Code-generation Pareto frontiers for Qwen2.5-7B-Instruct with the shared head-calibration procedure. The nine draft-head operating points in each panel use all three K_v columns of Table 3; adjacent numbers indicate K_v. NTP is the fine-tuned reference. Vertical and horizontal bars show sample SDs of pass@1 and per-run speedup, respectively. Dashed lines connect the nondominated measured means across all methods and K_v settings; hollow markers denote dominated points.

Table 11: Accuracy (%) by training duration. Differences in percentage points use unrounded scores.

Backbone	Task	MTP (2 epochs)	Dual-MTP (1 epoch)	Difference
Qwen2.5-1.5B	MATH500	20.80	23.80	+3.00
Qwen2.5-1.5B	GSM8K	48.75	49.28	+0.53
Qwen2.5-7B	MATH500	41.40	41.00	-0.40
Qwen2.5-7B	GSM8K	72.02	72.93	+0.91

I.2 Evaluation on Qwen3.5

We extend the mathematical-reasoning evaluation to the text decoders of Qwen3.5-4B and Qwen3.5-9B [27, 28]. All methods train for one epoch on the same task-specific corpora as in Appendix I.1.

Dual-MTP configurations were selected using task accuracy and throughput. For 4B/GSM8K, the heads also learn from the verifier's top-ranked tokens.

We report one full-test-set accuracy evaluation per configuration and measure speed on the first 32 questions of each task on an A100. Speedup is relative to fine-tuned NTP. On 9B/GSM8K, the NTP accuracy and timing results use different checkpoints.

Table 12: Qwen3.5 mathematical-reasoning results. NTP reports accuracy (%); the other methods report accuracy (%) / speedup relative to the task-matched NTP timing reference. Accuracy uses the full test set, while speedup uses a fixed 32-question subset.

Backbone	Task	NTP	MTP	Medusa	Dual-MTP
Qwen3.5-9B	GSM8K	43.97	68.23 / 1.368	58.61 / 1.638	**70.20** / 1.382
	MATH500	**50.40**	49.20 / 1.354	44.00 / 1.587	**50.40** / 1.327
Qwen3.5-4B	GSM8K	67.78	63.61 / 1.189	48.82 / 1.466	**70.05** / 1.207
	MATH500	43.20	39.60 / 1.283	42.20 / 1.490	**43.60** / 1.220

Dual-MTP attains the highest accuracy in three model–task settings and ties NTP on 9B/MATH500 at 50.40%. It exceeds MTP by 1.20–6.44 percentage points. Its measured speedup over the NTP timing reference ranges from 1.207× to 1.382×. Medusa has the highest throughput, while Dual-MTP has the highest or tied-highest accuracy.

I.3 Instruction Following on IFEval

We evaluate Qwen2.5-7B-Instruct on IFEval [43], using the complete set of 541 prompts and 834 verifiable instructions. Strict-prompt accuracy requires every instruction in a prompt to be satisfied; strict-instruction accuracy counts each instruction separately. Both metrics use the official evaluator.

All four methods are fine-tuned on the same 15,011 examples: 12,000 from Tulu-3 Persona Instruction-Following[2] and 3,011 from UltraChat `train_sft`[3]. We filter the training data for overlap with IFEval prompts and train for one epoch on assistant responses.

Table 13 reports both accuracy metrics and throughput for one full-set evaluation per method, with a limit of 2048 generated tokens. The draft-based methods use $K_v = 5$, and the NTP reference is fine-tuned on the same instruction data.

Table 13: Instruction following on IFEval with Qwen2.5-7B-Instruct. Accuracy is reported as percentage (correct/total), with throughput in tokens per second.

Method	Strict prompt	Strict instruction	tok/s
NTP	70.24 (380/541)	77.58 (647/834)	64.12
MTP	70.06 (379/541)	77.22 (644/834)	69.73
Medusa	69.32 (375/541)	77.46 (646/834)	**74.38**
Dual-MTP	**72.46 (392/541)**	**78.54 (655/834)**	70.42

Dual-MTP improves strict-prompt and strict-instruction accuracy over the fine-tuned NTP reference by 2.22 and 0.96 percentage points, respectively, with 9.82% higher throughput. Compared with MTP, the corresponding accuracy gains are 2.40 and 1.32 points. Medusa achieves higher throughput, while Dual-MTP achieves higher accuracy on both metrics.

J Broader Impact

This work has the potential to improve the efficiency and reliability of large language model inference by reducing error propagation in multi-token prediction, which can lower computational costs and energy consumption in large-scale deployments. Training with perturbed contexts aims to improve generation quality while retaining the draft-and-verify inference procedure. However, improving generation quality and efficiency may also facilitate the broader deployment of language models, which could amplify existing risks such as the spread of misinformation or misuse in automated content generation. Overall, the societal impact of this work largely inherits that of the underlying language models, without introducing fundamentally new risks, but may modestly influence the balance between efficiency, accuracy, and accessibility.

[2] `https://huggingface.co/datasets/allenai/tulu-3-sft-personas-instruction-following`
[3] `https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k`

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our claims have been reflected in both theoretical analysis and empirical results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section 8.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: Assumptions and proofs are given in Appendices A and B; the predictive-entropy approximation is established in Appendix D.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Experimental settings are described in Section 6. Implementation details and additional configurations are given in Appendices C, F, G, H, and I.

   Guidelines:

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

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We have provided an open-sourced link for our code, with a well-organized README file.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have provided the details in Section 6.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Means and standard deviations summarize repeated evaluations in Tables 1 and 2 and Appendices F and H. Table 7 and Appendix I report a single evaluation per configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix G reports training costs, and Appendix H describes how throughput and speedup are measured.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses only public open-source models and benchmarks, involves no human subjects or sensitive data, and conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We analyze the broader impacts of our work in Appendix J.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper releases no pre-trained models or scraped datasets and poses no high-risk safeguards concerns.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All base LLMs and benchmarks used in this work are properly cited in Section 6 and used within the terms of their public releases.

Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The README documents the Dual-MTP code, the only new asset.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: This paper only adopts LLMs for writing polish.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.