LIGHTMTP: LIGHTWEIGHT LATENT MULTI-TOKEN PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Next-token prediction (NTP) is the standard pretraining objective for large language models, yet it provides an explicit training signal only for the *immediate* next token, which can lead models to exploit local patterns instead of capturing longer-range structure and ideas. Multi-token prediction (MTP) addresses this by training models to predict several future tokens. However, existing MTP methods often introduce a large number of new parameters with limited improvements in downstream performance. Latent MTP approaches address this efficiency issue by encoding future tokens into a vector representation. However, these approaches usually rely on external helper models for future token encoding. We propose LightMTP, a lightweight, i.e., parameter-efficient, latent MTP approach that bootstraps the future token representations from the model's *own* hidden states. Our two LightMTP variants extend supervision to more future tokens without requiring the additional computational overhead of conventional MTP nor the external supervision latent MTP normally relies on. LightMTP adds at most 1% extra parameters, retains better performance on general language modeling benchmarks, and achieves similar gains in planning, coding, and reasoning.

1 INTRODUCTION

Next-token prediction (NTP) is the standard pretraining objective for large language models. It requires no supervision beyond raw text, computes the loss at every position in a single teacher-forced forward pass, and improves predictably with scale (Kaplan et al., 2020; Hoffmann et al., 2022). However, NTP provides an explicit training signal only for the immediate next token. Information about more distant tokens is learned only insofar as it helps predict the next one, which can lead models to exploit local patterns instead of capturing longer-range structure. Bachmann & Nagarajan (2024) show that teacher-forced NTP can fail to learn even simple planning tasks.

Multi-token prediction (MTP) addresses this limitation by additionally training the model to predict the tokens $x_{t+2}, \ldots, x_{t+k}$ from its representation at position t. This densifies the training signal and rewards representations that encode information beyond the next token (Gloeckle et al., 2024; DeepSeek-AI et al., 2025). Most MTP methods operate at the token level, attaching one prediction module per future offset that outputs a vocabulary distribution. However, this is costly: each offset adds a vocabulary projection, so compute and memory grow with both k and the vocabulary size, and many designs add highly parametrized transformer layers per offset. Additionally, token-level MTP can also interfere with next-token prediction. Gloeckle et al. (2024) find that it hurts models below roughly 3B parameters.

Latent MTP methods avoid the vocabulary projection by predicting dense representations of future tokens. They typically obtain their targets from auxiliary encoders or separately trained helper models (Yin et al., 2024; Mahajan et al., 2026). We propose **LightMTP, a lightweight latent MTP objective that bootstraps its targets from the model's own hidden states** (Figure 1). A small MLP head reads the last hidden state at position t and directly predicts the intermediate layer states of future positions up to k steps ahead. Because the targets are latent, the model is encouraged to encode information about upcoming tokens without committing to specific ones. LightMTP requires no auxiliary model, avoids projecting onto the vocabulary, and adds at most 1% parameters. The head is discarded after training, so inference remains unchanged.

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

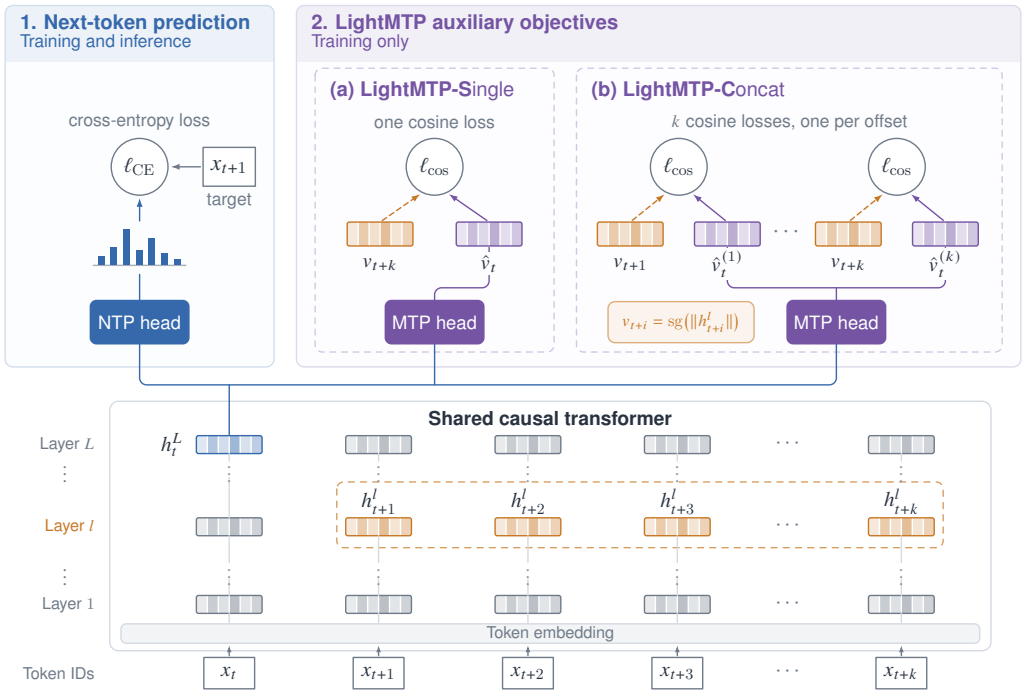


Figure 1: LightMTP passes the input through a shared transformer trunk and uses the last hidden state at each position to 1) perform next-token prediction via a linear NTP head, and 2) predict a latent representation of the next k target vectors stemming from the model's own future hidden states via an MTP head. a) In our approach's single-target variant, the model predicts the single hidden state vector in k steps. b) In the version with concatenated targets, each intermediate vector is additionally used to supervise the model.

We compare LightMTP with NTP, three token-level MTP methods, and a latent MTP baseline at model sizes up to 1.5B parameters. LightMTP produces hidden states that are substantially more predictive of future tokens than NTP, while incurring a smaller next-token performance cost than the MTP baselines. On Countdown, a planning task, LightMTP improves over NTP by up to 10 points and more often preserves target reachability. On code generation and math reasoning, it follows output formats more consistently and makes fewer structural errors than NTP. Unlike the token-level MTP baselines, LightMTP also retains NTP-level performance on general language modeling benchmarks, supporting its use as a general-purpose pretraining objective.

2 Related Work

NTP vs MTP. Predicting several future tokens predates decoder-only LLMs, having been explored both to accelerate decoding (Stern et al., 2018) and as a pretraining objective in ProphetNet (Qi et al., 2020) to reduce local overfitting. Gloeckle et al. (2024) first applied MTP to decoder-only transformer models by assigning each future token position its own head, with all positions sharing the unembedding matrix and transformer's trunk. For simplicity, their method does not fully model the joint probability of the next k tokens but approximates it by assuming conditional independence given the previous context. This led Basharin et al. (2025) to propose using a rank-r probability decomposition to better approximate the joint future token distribution, subsequently achieving lower joint token loss on one dataset. However, they do not demonstrate the impact on more general pretraining benchmarks. The first to pretrain a large-scale frontier model with MTP, DeepSeek-v3 models the joint token probability via sequential per-position modules that partially share parameters (DeepSeek-AI et al., 2025). However, they only train with one additional token ($k = 2$). While it has been shown that LLMs plan internally for some future tokens beyond the immediate next one (Pal et al., 2023), this effect is limited and might benefit from more explicit supervision during training.

Latent MTP. Latent MTP methods differ mainly in where their targets come from. Semformer (Yin et al., 2024) trains planning tokens to predict an autoencoder's representation of the response, next-concept prediction (Liu et al., 2026) predicts vector-quantized concepts spanning several tokens, and Future Summaries (Mahajan et al., 2026) predicts the hidden states of a separately pretrained right-to-left language model. These targets require auxiliary encoders, codebooks, or helper models, which add parameters and training stages. Similar to our work, NextLat (Teoh et al., 2026) also regress the model's own stop-gradient hidden states. They train a transition model that must predict each next hidden state from only the current state and the next token to obtain more compact, less shortcut-prone representations that generalize and plan better. LightMTP differs in three ways. It predicts states up to k steps ahead from the current state alone, without access to the intervening ground-truth tokens, learning to anticipate rather than model transitions. Furthermore, it predicts the future in a single parallel head call rather than a sequential rollout. Lastly, it takes its targets from an intermediate layer and uses only a regression loss, avoiding any projection onto the vocabulary.

3 LightMTP: Lightweight Latent Multi-Token Prediction

We now describe the architecture of LightMTP.

3.1 Latent Prediction Head and Training Objective

A transformer trunk with L layers maps a prefix $\mathrm{x}_{\leq t}$ to hidden states $\boldsymbol{h}_t^l$ for $l = 1, \ldots, L$. Standard NTP trains a linear head $\boldsymbol{M} \in \mathbb{R}^{|V| \times d}$ on $\boldsymbol{h}_t^L$ with a cross-entropy loss $\mathcal{L}_{\text{NTP}}$. As visualized in Figure 1, we add a second head $g_{\boldsymbol{\phi}}$, parametrized separately from the trunk parameters $\boldsymbol{\theta}$, which reads $\boldsymbol{h}_t^L$ and predicts hidden states the trunk itself produces at future positions. During training, this head shapes the transformer trunk's parameters that are shared with the NTP objective, leading to hidden states that are more informed about future tokens. In our experiments, $g_{\boldsymbol{\phi}}$ is a two-layer MLP with a GELU nonlinearity.

Targets. As target vectors, we consider the trunk's own hidden states $\boldsymbol{h}_\tau^l$ at a chosen target layer $0 \leq l \leq L$, taken at some future position $\tau > t$. As these are produced by the same trunk that is being trained, we stop gradients from flowing into $\boldsymbol{\theta}$ through the target branch, denoted $\mathrm{stopgrad}(\boldsymbol{h}_\tau^l)$; the head's own prediction error still backpropagates into $\boldsymbol{\theta}$ through $\boldsymbol{h}_t^L$, the input to $g_{\boldsymbol{\phi}}$.

Loss. We normalize both the predicted and the target vectors. Given a normalized predicted vector $\hat{\boldsymbol{v}} \in \mathbb{R}^d$ and a normalized, stop-gradient target $\boldsymbol{v} \in \mathbb{R}^d$, we consider a cosine loss function $\ell(\hat{\boldsymbol{v}}, \boldsymbol{v})$:

$$\ell_{\cos}(\hat{\boldsymbol{v}}, \boldsymbol{v}) = 1 - \frac{\hat{\boldsymbol{v}} \cdot \boldsymbol{v}}{\|\hat{\boldsymbol{v}}\|_2 \, \|\boldsymbol{v}\|_2} \tag{1}$$

Encouraging representational diversity in the target layer. Because the target layer's representations are themselves shaped by training, the model could potentially make all positions similar, trivializing the MTP loss without transferring useful information. While the NTP objective encourages some representational diversity, we additionally experiment with *Variance Regularization*, a within-sequence variant of cosine regularization (Gao et al., 2019) that penalizes cross-positional similarity of the normalized target-layer representations, computed per batch element $b \in \{1, \ldots, B\}$ over the sequence of length T:

$$\hat{\boldsymbol{h}}_{b,t}^l = \mathrm{norm}(\boldsymbol{h}_{b,t}^l), \qquad \bar{\boldsymbol{h}}_b^l = \frac{1}{T} \sum_{t=1}^{T} \hat{\boldsymbol{h}}_{b,t}^l, \qquad \mathcal{L}_{\text{VarReg}} = \frac{1}{B} \sum_{b=1}^{B} \|\bar{\boldsymbol{h}}_b^l\|_2 \tag{2}$$

A large mean-vector norm $\|\bar{\boldsymbol{h}}_b^l\|_2$ indicates that positions share a common direction rather than being spread out, so penalizing it encourages position-specific representations.

Overall loss. Both variants below are trained with

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{NTP}} + \lambda_{\text{MTP}} \, \mathcal{L}_{\text{MTP}} + \lambda_{\text{VarReg}} \, \mathcal{L}_{\text{VarReg}}, \tag{3}$$

where $\mathcal{L}_{\text{NTP}}$ is standard Cross-Entropy Loss and $\mathcal{L}_{\text{MTP}}$ is defined separately for each variant.

3.2 LIGHTMTP-S: SINGLE-TARGET VECTOR PREDICTION

The simplest instantiation of g_{ϕ} predicts a single future representation k steps ahead, rather than modeling every intermediate position. Concretely, $g_{\phi} : \mathbb{R}^d \to \mathbb{R}^d$ maps the last hidden state at position t to a prediction of the target-layer hidden state k steps in the future:

$$\hat{\boldsymbol{v}}_t = \mathrm{norm}\big(g_{\phi}(\boldsymbol{h}_t^L)\big), \qquad \boldsymbol{v}_{t+k} = \mathrm{norm}\big(\mathrm{stopgrad}(\boldsymbol{h}_{t+k}^l)\big). \tag{4}$$

The MTP loss averages the per-position loss over the sequence:

$$\mathcal{L}_{\text{MTP}}^{\text{single}} = \frac{1}{T-k} \sum_{t=1}^{T-k} \ell(\hat{\boldsymbol{v}}_t, \boldsymbol{v}_{t+k}). \tag{5}$$

Note that k here only sets the lookahead *offset* of a single target vector. It does not aggregate information from the intervening $k-1$ positions but relies on the vector's contextualized representation for intermediate information.

3.3 LIGHTMTP-C: CONCATENATED MULTI-TARGET VECTOR PREDICTION

Rather than committing to a single future point, we can instead task g_{ϕ} to *jointly* predict the hidden states for all k future positions by concatenating them along the channel dimension. Here $g_{\phi} : \mathbb{R}^d \to \mathbb{R}^{k \cdot d}$ produces a single $k \cdot d$-dimensional output, which we split into k length-d slices $\hat{\boldsymbol{v}}_t^{(1)}, \ldots, \hat{\boldsymbol{v}}_t^{(k)}$, each compared against its corresponding future hidden state:

$$[\hat{\boldsymbol{v}}_t^{(1)}; \ldots; \hat{\boldsymbol{v}}_t^{(k)}] = g_{\phi}(\boldsymbol{h}_t^L) \in \mathbb{R}^{k \cdot d}, \tag{6}$$

$$\hat{\boldsymbol{v}}_t^{(i)} \leftarrow \mathrm{norm}\big(\hat{\boldsymbol{v}}_t^{(i)}\big), \qquad \boldsymbol{v}_{t+i} = \mathrm{norm}\big(\mathrm{sg}(\boldsymbol{h}_{t+i}^l)\big), \qquad i = 1, \ldots, k. \tag{7}$$

Each slice is normalized independently, so the loss scale does not depend on k. We compute one loss term per future step i, averaged over positions t that are valid for that step, i.e., where $t + i$ remains within the same packed document and before the sequence ends ($m_{t,i} = 1$; this lets end-of-sequence positions still train on whichever of their k future steps are in range, instead of discarding them entirely):

$$\ell_i = \frac{\sum_t m_{t,i}\, \ell\big(\hat{\boldsymbol{v}}_t^{(i)}, \boldsymbol{v}_{t+i}\big)}{\sum_t m_{t,i}}. \tag{8}$$

The k per-step losses are then aggregated, using uniform weights, as

$$\mathcal{L}_{\text{MTP}}^{\text{concat}} = \sum_{i=1}^{k} \ell_i, \tag{9}$$

targeting the same k future positions as k applications of the single-target head at offsets $1, \ldots, k$.

4 EXPERIMENTAL SETUP

We investigate the impact of latent MTP via four avenues: 1) models' hidden states, 2) general language modeling performance, 3) models' local planning abilities and, 4) structural tasks via coding and reasoning.

Models. We pretrain 24-layer decoder-only models with 1.4–1.5B parameters (Table 6), a scale at which MTP has often failed to help. We compare five baselines, namely NTP, two MTP variants in the style of Gloeckle et al. (2024) from Meta[1], DeepSeek-style MTP, and Future Summaries (see §A.2 for details) with the two LightMTP variants, LightMTP-S(ingle) and LightMTP-C(oncat). Unless stated otherwise, LightMTP uses target layer $l = 16$ (two-thirds of depth; we use the same ratio at depths 12 and 18) and horizon $k = 4$, following related work. We also report $k = 10$ in Table 1, and a full horizon sweep in Appendix B.7. To isolate the effect of the variance regularizer, we train every LightMTP variant with and without VarReg ($\lambda_{\text{VarReg}} = 0.1$). In the probing analysis (§5.1), we also include target layer $l = 2$ to test whether VarReg's effect on anticipation holds across target layers. We do not use $l = 2$ for downstream evaluation. Since NTP is the essential baseline, it is trained over five seeds, all other models over three.

[1]We denote "Meta" as the paper's reported implementation with an extra Transformer layer per MTP head, and "Meta-L" as their linear variant described in the paper's appendix.

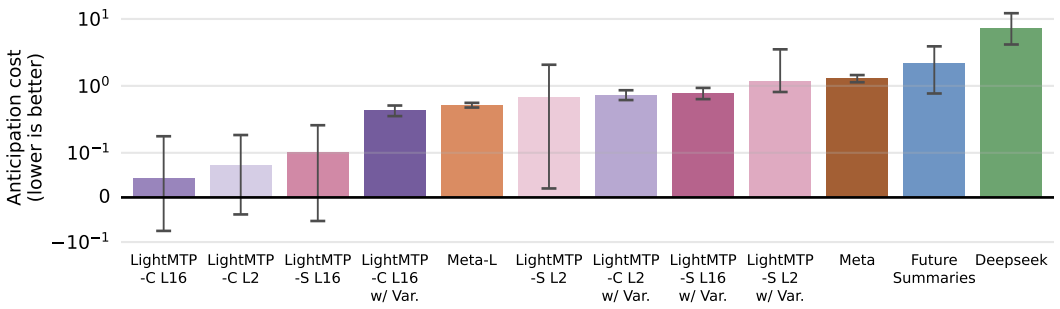


Figure 2: **LightMTP pays fewer bits-per-byte –** Plot of anticipation cost, which quantifies how many bits-per-byte the models pay per total future token prediction improvement up to position k for $k \leq 10$ (lower is better). The top four most cost-efficient models are LightMTP variants.

Pretraining. We pretrain models with a parameter-dependent token budget. (Results for models with 12 and 18 layers, respectively, are reported in Appendices C and C.2.) This ensures that Meta's and DeepSeek's approaches, which add a substantial amount of MTP parameters, are not undertrained compared to the other models, but it leads to them seeing +20.7%/+11.6% (Meta-L/Meta) and +13.6% (DeepSeek) more tokens, respectively. We use a global batch size of $524{,}288$ tokens and a tokenizer of size $32{,}768$. For an overview of the model's architecture, used hyperparameters, and parameter and token budgets, see Appendix A.

5 Results

We organize the results around the cost of better future token representation (§ 5.1), general language modeling benchmarks (§ 5.2), planning (§ 5.3), and reasoning and coding (§ 5.4) abilities.

5.1 Future Token Anticipation

LightMTP, as well as the other MTP approaches, adds an explicit local lookahead pressure to the pretraining objective. To measure how this additional objective affects the models' representations, we train linear probes to predict the token at $t + k$ using the models' frozen last hidden-layer outputs on a held-out FineWeb-Edu shard. Since the last hidden layer directly receives gradients from the MTP heads, we expect MTP models to surface more future-token information in this layer, making future tokens more easily linearly decodable. The resulting probing results, shown in Appendix B.1, reveal a tradeoff: **MTP methods generally improve future-token prediction ($k \geq 2$), but this comes at a cost to NTP ($k = 1$)**, most prominently for Meta's and DeepSeek's MTP approaches. Variance-based regularization further improves future-token anticipation for LightMTP, but also increases this cost.

To quantify this tradeoff, we introduce *anticipation cost*, defined as the degradation in NTP divided by the gain in future-token prediction:

$$\text{Anticipation Cost} = \frac{\Delta\text{Bpb}_{k=1}}{\Delta\text{Acc}_{k>1}}.$$

Here, $\Delta\text{Bpb}_{k=1}$ is the increase in next-token bits-per-byte relative to NTP, while $\Delta\text{Acc}_{k>1}$ is the corresponding gain in future-token prediction accuracy. Thus, anticipation cost captures how much next-token performance a method gives up for each unit of additional future-token information encoded in its representations. Figure 2 visualizes anticipation cost across MTP methods. Four LightMTP variants (LightMTP-C L16 and L2, LightMTP-S L16, and LightMTP-C L16 with variance regularization) exhibit the lowest anticipation cost of all methods, with Meta-L in fifth place. Future Summaries and DeepSeek's approach have anticipation costs two orders of magnitude higher than the best LightMTP variant. Overall, these results suggest that, at this model scale, increasing future-token anticipation eventually comes at the cost of NTP quality, and that this tradeoff may be a key driver of the general language modeling impairment observed for baseline MTP methods in Section 5.2. Notably, **LightMTP achieves substantially more future-token prediction with much smaller NTP**

Method	FineWeb-Edu (PPL)	Lambada (PPL)	Lambada (Acc)	CORE-avg
NTP	**1.683**$_{\pm 0.001}$	16.00$_{\pm 0.72}$	43.59$_{\pm 0.85}$	<u>40.43</u>$_{\pm 0.68}$
Meta	1.712$_{\pm 0.003}$	18.66$_{\pm 0.84}$	42.14$_{\pm 0.91}$	39.31$_{\pm 0.69}$
Meta-L	1.698$_{\pm 0.001}$	18.13$_{\pm 0.48}$	42.84$_{\pm 0.70}$	39.89$_{\pm 0.42}$
DeepSeek	1.762$_{\pm 0.038}$	31.49$_{\pm 11.42}$	35.68$_{\pm 4.28}$	35.81$_{\pm 2.60}$
Future Summaries	1.685$_{\pm 0.002}$	**15.57**$_{\pm 0.33}$	**44.43**$_{\pm 0.45}$	40.25$_{\pm 0.37}$
LightMTP-C	**1.683**$_{\pm 0.000}$	16.35$_{\pm 0.20}$	43.22$_{\pm 0.42}$	40.30$_{\pm 0.23}$
+VarReg	1.689$_{\pm 0.000}$	16.84$_{\pm 0.41}$	42.65$_{\pm 0.51}$	40.18$_{\pm 0.43}$
k=10	1.685$_{\pm 0.005}$	15.97$_{\pm 1.04}$	<u>44.06</u>$_{\pm 1.47}$	**40.57**$_{\pm 0.34}$
LightMTP-S	<u>1.684</u>$_{\pm 0.000}$	16.28$_{\pm 0.48}$	43.31$_{\pm 0.71}$	40.13$_{\pm 0.67}$
+VarReg	1.694$_{\pm 0.002}$	17.53$_{\pm 0.35}$	42.33$_{\pm 0.14}$	40.29$_{\pm 0.46}$
k=10	1.685$_{\pm 0.001}$	<u>15.92</u>$_{\pm 0.46}$	43.97$_{\pm 0.16}$	40.19$_{\pm 0.34}$

Table 1: **LightMTP retains or even improves general LM benchmark performance –** perplexity on FineWeb-Edu and Lambada, and accuracy on Lambada and CORE, each result averaged over three seeds. FineWeb-Edu results report byte-level perplexity 2^{bpb} on a held-out shard. The Lambada columns show the FineWeb-Edu-pretrained *base* model applied to Lambada's validation split, measuring perplexity and accuracy at the target word. In each column, the best value is **bolded** and the runner-up <u>underlined</u>.

degradation, suggesting that its intermediate-layer supervision provides a more favorable tradeoff between anticipation and next-token modeling.

5.2 Language Modeling Performance: LightMTP Shows Least Degradation

Commonly used to evaluate models' reasoning, world knowledge, and long-range capabilities, we report performance on the CORE task suite (Li et al., 2025) and LAMBADA[2] (Paperno et al., 2016). CORE comprises 22 datasets spanning a broad range of language model capabilities. We include LAMBADA, since it tests long-range context use through a cloze task, in which the final word of a short story can only be predicted by incorporating context beyond the final sentence.

Prior MTP work found that language modeling performance degrades under MTP objectives. Our results confirm this for the three token-level MTP baselines: both Meta variants and DeepSeek MTP regress relative to NTP across every metric, with DeepSeek MTP showing the most severe degradation. Meta-L degrades less severely but still consistently underperforms NTP on every column.
LightMTP, by contrast, shows almost no degradation and in several cases improves over NTP. On FineWeb-Edu perplexity, LightMTP-C L16 matches NTP exactly, and most other LightMTP variants stay within 0.01 of the NTP baseline. On CORE-avg, LightMTP-C with $k = 10$ achieves the best score overall, with the default $k = 4$ slightly behind NTP. LightMTP-S L16 (k=10) achieves the second-best Lambada perplexity in the table, only edged out by Future Summaries. This result aligns with Future Summaries' training on long-horizon targets obtained from their auxiliary right-to-left-trained target model. Notably, Future Summaries also avoids the degradation seen in the MTP baselines, though it trails both LightMTP variants as well as NTP on CORE-avg.
Although the variance-regularized (+VarReg) variants improved future-token anticipation in probing experiments, VarReg consistently harms general language modeling capabilities relative to their non-regularized LightMTP counterparts. Both +VarReg variants show worse FineWeb-Edu perplexity than their base counterpart, and the degradation continues on Lambada perplexity and accuracy. This divergence between probing-level gains and downstream degradation suggests gradient interference or representational competition between the future-token-anticipation objective and NTP, an explanation consistent with variance regularization sharpening the former at the latter's expense.

[2]LAMBADA is also included in the CORE suite.

5.3 PLANNING: COUNTDOWN

We expect MTP training to improve planning, as it encourages models to consider longer horizons. In *Countdown* (Gandhi et al., 2024), players combine four numbers via basic arithmetic operators (+, -, ×, ÷) to arrive at a target number. Countdown requires models to explore the search space spanned by the numbers and operators, with which NTP has been shown to struggle. We follow Teoh et al. (2026) to generate data where targets $\in [10, 100]$. We define accuracy as the ratio of arithmetically correct answers that consume each input number exactly once and end on the target.[3]

On the held-out Countdown targets, **both LightMTP variants substantially outperform the NTP baseline and improve consistently as the lookahead horizon k increases**. LightMTP-S outperforms LightMTP-C at every k and does so with lower variance at the largest horizon. Notably, it achieves this with only 1.2M MTP parameters, compared to 75M for the Meta-L objective: a 98% reduction in parameter count. Since for $k{=}1$ the Meta models are identical to an NTP model, they cannot offer gains at this horizon. Only Meta-L surpasses LightMTP at $k \geq 4$, reaching its own peak at $k{=}8$. The DeepSeek MTP objective peaks earlier, at $k{=}4$, then *declines* at $k{=}8$, the only method for which additional lookahead is counterproductive. Despite its smaller footprint, LightMTP-S trails Meta's best result by only 2.2 points (56.7% vs. 58.9%) while matching it in stability and monotonicity, suggesting a more favorable accuracy-per-parameter trade-off. The Future Summaries baseline, by contrast, underperforms NTP on average. However, this is due to held-out accuracy destabilizing late in training on two of three seeds. Together these results indicate that LightMTP extracts benefit from longer horizons more reliably than the DeepSeek and Future Summaries objectives, without the instability or non-monotonicity seen in these baselines, and with a fraction of Meta's parameters.

To test whether MTP models' gains reflect better planning, we examine the solution space remaining after each generated equation. Models that plan ahead should preserve more paths to the target, while greedy models are expected to more often enter dead ends. Figure 3 shows the fraction of solutions remaining after each equation step. LightMTP-C and -S, as well as Meta and DeepSeek, preserve more solutions than NTP throughout the generation. This suggests that these **MTP methods do indeed plan ahead**: their first equations are not only arithmetically correct, but also tend to preserve a larger set of paths to the target. Future Summaries is an outlier: it preserves more solutions than NTP initially, but degrades over subsequent steps, resulting in lower final performance.

5.4 STRUCTURE: GSM8K AND MBPP

Because MTP explicitly models immediate future tokens, we expect MTP models to exhibit better local look-ahead and better adherence to surface structure in generative tasks. We evaluate performance on GSM8K, a grade-school reasoning benchmark (§ 5.4), and MBPP, a Python coding dataset (§ 5.4). We further analyze answer-format adherence and failure cases.

Reasoning on GSM8K. GSM8K (Grade School Math 8K; Cobbe et al., 2021) is a benchmark evaluating models on math and reasoning. The examples require basic arithmetic and compositional skills. We evaluate the base models only pretrained on FineWebEdu without additional finetuning on GSM8K's train set. At inference, we include eight train-set in-context examples to encourage the models to pick up the answer format. We present results in Table 3. The *relaxed* metric checks whether the gold answer exists anywhere in the generated output, while the *strict* metric only counts answers given in the format of "The answer is [ANSWER]" or "#### [ANSWER]". Since a model could potentially answer perfectly but without following the answer format we visualize the ratio of correct answers to format following in Figure 4.

On GSM8K, **LightMTP variants outperform NTP** on the relaxed metric across both configurations, **while the Meta and DeepSeek MTP baselines regress below NTP** on both metrics. Under the strict metric, where answers must follow the exact required format, LightMTP-S achieves the highest score, followed by LightMTP-C, essentially tied with NTP. Future Summaries again matches NTP-level performance. Both Meta variants and DeepSeek MTP underperform NTP on strict accuracy, despite being trained on 11.6%, 20% and 13% more tokens, respectively. This reinforces that their regressions are not an artifact of the relaxed metric.

[3]Note that since our models are built on top of the `nanochat` repository, the underlying transformer backbone is not identical to Teoh et al. (2026) and so their numbers are not directly comparable to ours.

Objective	Held-out targets
NTP	46.4 ± 1.4
Meta, k=4	51.4 ± 3.1
Meta, k=8	58.1 ± 0.5
Meta-L, k=4	55.8 ± 0.9
Meta-L, k=8	58.9 ± 1.2
DeepSeek, k=1	47.8 ± 2.5
DeepSeek, k=4	55.4 ± 0.4
DeepSeek, k=8	55.1 ± 0.3
Future Summaries‡	34.9 ± 15.5
LightMTP-C, k=1	50.1 ± 2.6
LightMTP-C, k=4	54.0 ± 3.0
LightMTP-C, k=8	56.1 ± 1.4
LightMTP-S, k=1	50.6 ± 0.9
LightMTP-S, k=4	55.5 ± 1.2
LightMTP-S, k=8	56.7 ± 0.9

‡ held-out strict accuracy destabilizes late in training on 2 of 3 seeds (39.2 / 17.7 / 47.7).

Table 2: **MTP models better plan ahead –** Countdown strict accuracy results on held-out targets (%), mean ± s.d. over three seeds, greedy decoding. Shading reflects the gain over NTP.

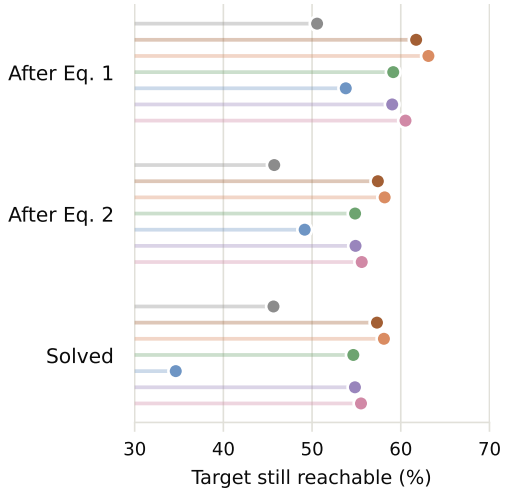


Figure 3: **MTP retains more paths to the solution –** Plot of where the solution space is lost. Dots mark the share of held-out problems on which the target is still reachable after each equation. Colors match the table. Best k per method.

Method	p@5 relaxed	p@5 strict
NTP	41.97 ±2.29	4.94 ±1.44
Meta	41.45 ±1.43	4.09 ±0.08
Meta-L	41.91 ±0.88	4.82 ±1.19
DeepSeek MTP	39.17 ±2.11	3.85 ±0.68
Future Summaries	41.19 ±0.53	4.95 ±1.25
LightMTP-C	<u>43.62</u> ±1.51	<u>4.98</u> ±0.91
LightMTP-S	**43.82** ±1.33	**5.69** ±1.09

Table 3: **LightMTP is best on GSM8K –** GSM8K results for NTP, MTP baselines and LightMTP. 1) "relaxed" considers whether the gold answer appears anywhere in the output. 2) "strict" only counts answers given in the correct answer format. Best result in **bold**, second-best <u>underlined</u>.

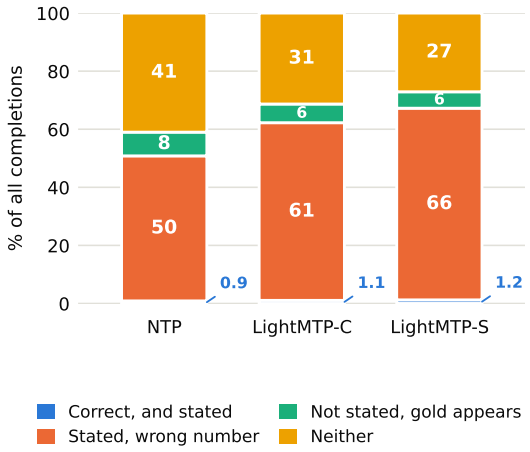


Figure 4: **LightMTP better adheres to answer structure –** The LightMTP models state the answer format more often than the NTP model, suggesting that LightMTP better picks up the format from the eight in-context examples.

Coding on MBPP. Since code is more structured than natural language, we posit that it will especially benefit from MTP's improved local look-ahead capabilities. Since FineWeb-Edu contains almost no code, we sample code data from The Stack (Kocetkov et al., 2022) V1.2's Python subset, and, for these experiments only, mix it into the pretraining dataloader so that about 10% of the final data mixture is Python code. Then, we fine-tuned the pretrained models on MBPP's (Austin et al., 2021) training set. We report results on the open validation set in Table 4.

Method	zero-shot		finetuned	
	p@5	p@10	p@5	p@10
NTP	5.02 ±0.97	8.52 ±1.70	11.39 ±1.22	15.19 ±2.31
Meta	4.80 ±1.80	7.04 ±2.31	13.22 ±2.64	17.78 ±3.85
Meta-L	6.01 ±2.10	9.26 ±2.80	**16.43** ±0.48	21.85 ±0.64
DeepSeek	0.19 ±0.32	0.37 ±0.64	2.47 ±1.90	3.70 ±2.57
Fut. Sum.	6.57 ±0.63	8.89 ±1.11	15.62 ±1.14	21.11 ±1.11
LightMTP-C	**6.94** ±0.10	**9.63** ±0.64	16.42 ±1.18	**23.33** ±2.94
LightMTP-S	5.30 ±2.28	8.89 ±4.44	12.55 ±1.63	17.78 ±2.22

Table 4: **LightMTP-C outperforms on code generation –** Code-generation pass@k (%) for the d24 python-mix models (10% Python in the pretraining mix) on MBPP. All models trained with $k = 4$. Bold marks the best mean per column. All results reported as mean ± std over three seeds.

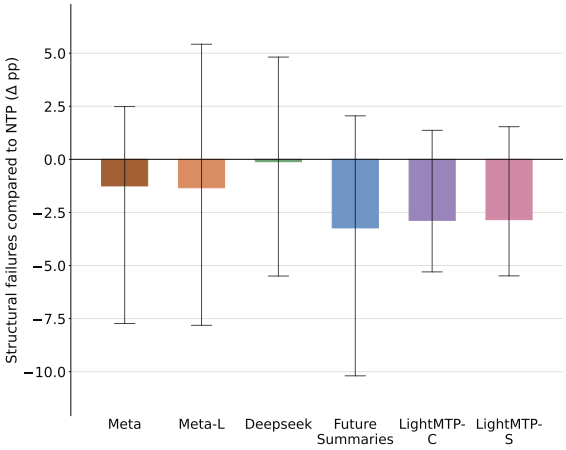


Figure 5: **MTP reduces structural errors –** Structural error rate of failed samples compared to NTP. Results over three seeds, with error bars indicating maximum and minimum scores. All MTP methods exhibit a lower structural error rate than NTP.

In the zero-shot MBPP setting, LightMTP-C achieves the best performance at both pass@5 and pass@10, improving over NTP and all four MTP baselines. After fine-tuning on MBPP, Meta-L takes the lead on pass@5, with LightMTP-C effectively tied and ahead on pass@10. LightMTP-S improves on NTP for finetuned MBPP pass@5 but performs well below LightMTP-C and Meta-L. DeepSeek MTP collapses on this task, scoring near zero in a zero-shot regime and remaining far behind even after finetuning, indicating that the objective transfers poorly to code.

To further investigate where MTP's edge comes from, we quantify the models' error types in Figure 5. Each failing completion was classified by the raised exception type. We report *structural errors*, consisting of `NameError`, `SyntaxError`, `IndentationError`, and `ModuleNotFoundError`, i.e., failures that are attributable to local naming/syntax consistency rather than task or data understanding.

Compared to NTP, all MTP models exhibit fewer structural errors, to a minimal degree even DeepSeek, which underperformed NTP on task performance by a large margin. Fewer structural errors support our hypothesis that MTP objectives teach *local* look-ahead skills, thus benefiting local coherence but not necessarily underlying task understanding. Future Summaries benefits from the largest reduction in structure-related errors, followed by the two LightMTP variants, and a more moderate effect on Meta and Meta-L. For some qualitative examples of structure-related generation differences between NTP and LightMTP, see Appendix B.6.

6 Conclusion

This paper introduces LightMTP, a latent multi-token prediction objective using a model's own future hidden states for supervision via a small auxiliary head. By predicting in latent space rather than over the vocabulary, it avoids the compute and memory overhead of token-level MTP, adding at most 1% extra parameters and leaving the trunk and the inference path untouched. **Across probing, planning, reasoning, and code generation, LightMTP balances between anticipating future tokens and retaining next-token performance better** than all considered MTP baselines.

This balance is particularly relevant at smaller model scales ($< 3B$), a regime for which prior work either reports degraded language modeling or that it omits entirely: LightMTP improves at modeling local task structure while leaving general language modeling benchmarks largely unaffected, making it practical as a pretraining objective where existing MTP methods are not. Future work characterizing the choices affecting the anticipation cost systematically, across target granularity, prediction horizon, and model scale, would provide a principled basis for designing multi-token objectives rather than selecting them empirically.

ETHICS STATEMENT

This work proposes a pretraining objective for language models and evaluates it on publicly available data. It involves no human subjects, crowdsourcing, or personally collected data. We pretrain on FineWeb-Edu, a filtered subset of CommonCrawl, and, for the coding experiments, on the Python subset of The Stack v1.2, which contains only permissively licensed code and honors developer opt-out requests. We evaluate on established public benchmarks (CORE, LAMBADA, GSM8K, MBPP) and on synthetically generated Countdown problems. Models trained on web-scale corpora may inherit the biases, factual errors, and harmful content in that data. We did not apply safety tuning or bias mitigation beyond the filtering already performed by the dataset creators. Our models are small ($\leq$1.5B parameters), are not instruction-tuned, and were trained solely for research, so we consider their potential for misuse to be low. Training the models reported here required a substantial number of GPU hours, and we report all training budgets in Appendix 6 to support reproducibility.

AI USE STATEMENT

In this work, we used generative AI tools for code generation to support the implementation of our methods. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis, or interpreting results, and the following are not applicable to this work: generating synthetic data, formulating mathematical claims, providing critical ingredients for proving mathematical claims sets, assisting in the writing of proofs, proposing or refining hypotheses, assisting with translation, cleaning and reformatting datasets. Additionally, we used generative AI tools for support on scientific figure creation, to help draft smaller paper sections and improve our paper's readability as well as identify relevant related work. We have reviewed all AI-assisted work via extensive testing (for coding), and a manual verification of paper and claim existence in the original papers (for paper writing and related work identification). We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY

All models in this paper are trained from scratch on publicly available data and built on top of `nanochat`, a publicly available codebase. The NTP base architecture, optimizer and hyperparameter settings all follow `nanochat`, without modifications. All MTP-specific alterations for every model variant and depth are fully described in the main text and in Table 6. Our reimplementations of the MTP baselines, which deviate from the published architectures in the ways noted there, are described in Appendix A.2. Appendix B documents the preparation of every dataset we use and complete results. All data sources are public and all generated data is produced from fixed seeds. Every reported cell is a mean over three pretraining seeds, with standard deviations reported throughout.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, August 2021. URL http://arxiv.org/abs/2108.07732. arXiv:2108.07732 [cs.PL].

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction, March 2024. URL http://arxiv.org/abs/2403.06963. arXiv:2403.06963 [cs].

Artem Basharin, Andrei Chertkov, and Ivan Oseledets. Faster Language Models with Better Multi-Token Prediction Using Tensor Decomposition, February 2025. URL http://arxiv.org/abs/2410.17765. arXiv:2410.17765 [cs.LG].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John

Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. DeepSeek-V3 Technical Report, February 2025. URL http://arxiv.org/abs/2412.19437. arXiv:2412.19437 [cs].

Kanishk Gandhi, Denise H J Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah Goodman. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkEYojRqtm.

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*, volume 235 of *ICML'24*, pp. 15706–15734, Vienna, Austria, July 2024. JMLR.org.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training Language Models With Pause Tokens. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/76917808731dae9e6d62c2a7a6afb542-Paper-Conference.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan

Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL https://arxiv.org/abs/2001.08361v1.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The stack: 3 tb of permissively licensed source code. *Preprint*, 2022.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. DataComp-LM: In search of the next generation of training sets for language models, April 2025. URL http://arxiv.org/abs/2406.11794. arXiv:2406.11794 [cs.LG].

Yuliang Liu, Yunchong Song, Yixuan Wang, Kewen Ge, Alex Lamb, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Next Concept Prediction in Discrete Latent Space Leads to Stronger Language Models, February 2026. URL http://arxiv.org/abs/2602.08984. arXiv:2602.08984 [cs].

Divyat Mahajan, Sachin Goyal, Badr Youbi Idrissi, Mohammad Pezeshki, Ioannis Mitliagkas, David Lopez-Paz, and Kartik Ahuja. Beyond Multi-Token Prediction: Pretraining LLMs with Future Summaries, March 2026. URL http://arxiv.org/abs/2510.14751. arXiv:2510.14751 [cs].

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future Lens: Anticipating Subsequent Tokens from a Single Hidden State. In Jing Jiang, David Reitter, and Shumin Deng (eds.), *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-1.37. URL https://aclanthology.org/2023.conll-1.37/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

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 Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2401–2410, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.217. URL https://aclanthology.org/2020.findings-emnlp.217/.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Proceedings of the 32nd International Conference on Neural Information*

Processing Systems, NIPS'18, pp. 10107–10116, Red Hook, NY, USA, 2018. Curran Associates Inc.

Jayden Teoh, Manan Tomar, Kwangjun Ahn, Edward S. Hu, Tim Pearce, Pratyusha Sharma, Akshay Krishnamurthy, Riashat Islam, Alex Lamb, and John Langford. Next-Latent Prediction Transformers Learn Compact World Models, June 2026. URL http://arxiv.org/abs/2511.05963. arXiv:2511.05963 [cs.LG].

Yongjing Yin, Junran Ding, Kai Song, and Yue Zhang. Semformer: Transformer Language Models with Semantic Planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18669–18680, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1039. URL https://aclanthology.org/2024.emnlp-main.1039.

A SETUP

A.1 ARCHITECTURE AND HYPERPARAMETERS

All models build on the `nanochat` codebase, a compact reimplementation of a modern GPT-style decoder. We keep its architecture and training recipe unchanged across all methods and did not tune any hyperparameter for LightMTP or the baselines. Table 5 lists the settings for the 24-layer models; the 12- and 18-layer models differ only in depth and in the depth-derived quantities (width, heads, weight decay, training steps).

Architecture. Width scales with depth at a fixed aspect ratio ($d_{\text{model}} = 64 \cdot L$) with 128-dimensional attention heads. Each block is a pre-norm Transformer block with parameter-free RMSNorm, no bias terms, rotary position embeddings (base 10^4) followed by QK-normalization, and a $4\times$ MLP with a squared-ReLU activation. The token embedding is normalized before the first block and the final hidden state before the output head. Input embedding and output head are untied, and output logits are soft-capped as $15\tanh(z/15)$. Attention alternates between local windows ("S", half the context) and full context ("L") in a repeating "SSSL" pattern, with the last layer always using full context. Two further additions are learnable per-layer scalars that rescale the residual stream and mix the normalized input embedding x_0 back in, $x \leftarrow \lambda_l^{\text{res}} x + \lambda_l^0 x_0$, and ResFormer-style value embeddings: in every other layer (always including the last), a separate token-embedding table is added to the attention values through an input-dependent gate.

Optimization. Following `nanochat`, all 2D weight matrices inside the Transformer blocks are optimized with Muon, using Nesterov momentum, Polar-Express orthogonalization (5 iterations), per-neuron second-moment normalization, and cautious weight decay. Embeddings, value embeddings, the output head and the per-layer scalars are optimized with AdamW. The learning rate is held constant for the first half of training and then decays linearly to zero. Muon momentum warms up from 0.85 to 0.95 over the first 300 steps, and weight decay decays linearly to zero over training. Training uses bfloat16 autocast (logits and loss in fp32), `torch.compile`, and FlashAttention-3 on Hopper GPUs. The auxiliary heads of all MTP methods are 2D matrices and are therefore optimized with Muon under the same settings as the trunk.

Data and tokenization. We use `nanochat`'s byte-level BPE tokenizer with a vocabulary of 2^{15}=32,768 tokens and a GPT-4-style pre-tokenization pattern (digits grouped in runs of at most two), trained on 2B characters of FineWeb-Edu. Sequences of 2,048 tokens are packed with a BOS-aligned best-fit strategy: every row starts at a document boundary and is filled without padding by the largest documents that fit, cropping one document to fill any remaining space. Attention is causal over the packed row, while the LightMTP targets are masked so that they never cross a document boundary (Section 3.3).

Training budget. Following `nanochat`'s speedrun recipe, the number of optimizer steps is set so that the number of training tokens is 12 times the number of scaling parameters (Transformer matrices plus output head; embeddings excluded). The auxiliary heads count as scaling parameters, so methods with larger heads train for more steps (Table 6).

A.2 BASELINES SETUP

To make the comparison between LightMTP and the baseline MTP approaches fairer, we adjusted their implementations from the original works.

We report two token-level MTP baselines derived from Gloeckle et al. (2024), denoted **Meta** and **Meta-L**. Both follow the same recipe: a shared transformer trunk produces a hidden state at position t, from which k heads predict the tokens $x_{t+1}, \ldots, x_{t+k}$. They differ only in what a head *is*.

Meta (default). Each of the $k-1$ auxiliary heads is a single transformer layer applied to the trunk's final hidden state, followed by the shared unembedding, exactly as in the architecture Gloeckle et al. (2024) propose in the main body of their paper. We deviate from their design in one aspect: the $k=1$ head is a *linear projection only*, i.e., the shared unembedding matrix is applied directly to the

Architecture (d24)	
Layers / width / heads	24 / 1536 / 12 (head dim. 128, no GQA)
MLP	$4\times$ (6,144), ReLU^2, no biases
Normalization	parameter-free RMSNorm (pre-norm, embedding, final, QK)
Position encoding	RoPE, base 10^4
Attention windows	SSSL pattern (S = 1,024, L = 2,048 tokens), last layer L
Context length	2,048
Vocabulary	32,768, untied input/output embeddings
Logit soft-cap	$15\tanh(z/15)$
Extras	value embeddings (12 layers), per-layer λ^{res}, λ^0 scalars
Optimization	
Batch size	2^{19}=524,288 tokens
Muon (block matrices)	Learning rate 0.02, momentum $0.85\rightarrow0.95$ (300 steps), Nesterov, 5 Polar-Express steps, cautious weight decay $0.2\cdot(12/L)^2=0.05$ as linear decay
AdamW (embeddings, value emb.)	Learning rate $0.3\cdot(d_{\text{model}}/768)^{-1/2}\approx0.212$
AdamW (output head)	Learning rate $0.004\cdot(d_{\text{model}}/768)^{-1/2}\approx0.0028$
AdamW (scalars)	Learning rate 0.005 (λ^{res}), 0.5 (λ^0)
AdamW betas / eps / wd	(0.8, 0.95), λ^0: (0.96, 0.95) / 10^{-10} / 0
LR schedule	no warmup; constant, then linear decay to 0 over the last 50%
Gradient clipping	none
Precision	bfloat16 autocast, fp32 logits and loss
Token budget	$12\times$ scaling parameters (Table 6)

Table 5: Backbone architecture and training hyperparameters for the 24-layer models, inherited unchanged from `nanochat`.

Model	Total	Scaling	Non-scaling	MTP head	Δ params	Tokens	Steps	H100 GPU-h
d12 $768\times12\times6$								
NTP	286.3M	110.1M	176.2M	—	—	1.32B	2 520	0.7
Meta-L‡	361.8M	185.6M	176.2M	75.5M	+26.4%	2.23B	4 248	2.0
Meta	307.5M	131.3M	176.2M	21.2M	+7.4%	1.58B	3 006	1.6
DeepSeek MTP	311.0M	134.9M	176.2M	24.8M	+8.7%	1.62B	3 087	1.6
Future Summaries	287.4M(+286.3M)	111.3M	176.2M	1.2M	+0.4%	1.34B	2 547	0.9
LightMTP-single	287.4M	111.3M	176.2M	1.2M	+0.4%	1.34B	2 547	0.7
LightMTP-concat	289.2M	113.1M	176.2M	2.9M	+1.0%	1.36B	2 587	0.8
d18 $1152\times18\times9$								
NTP	701.9M	324.4M	377.5M	—	—	3.89B	7 425	5.4
Meta-L‡	815.1M	437.7M	377.5M	113.2M	+16.1%	5.25B	10 017	9.6
Meta	749.7M	372.2M	377.5M	47.8M	+6.8%	4.47B	8 518	9.1
DeepSeek MTP	757.6M	380.2M	377.5M	55.7M	+7.9%	4.56B	8 700	9.3
Future Summaries	704.5M(+701.9M)	327.1M	377.5M	2.7M	+0.4%	3.92B	7 485	7.0
LightMTP-single	704.5M	327.1M	377.5M	2.7M	+0.4%	3.92B	7 485	5.5
LightMTP-concat	708.5M	331.0M	377.5M	6.6M	+0.9%	3.97B	7 576	5.7
d24 $1536\times24\times12$								
NTP	1.38B	729.8M	654.3M	—	—	8.76B	16 704	24.5
Meta-L‡	1.54B	880.8M	654.3M	151.0M	+10.9%	10.57B	20 160	37.9
Meta	1.47B	814.7M	654.3M	84.9M	+6.1%	9.78B	18 648	37.9
DeepSeek MTP	1.48B	828.9M	654.3M	99.1M	+7.2%	9.95B	18 972	37.4
Future Summaries	1.39B(+1.38B)	734.5M	654.3M	4.7M	+0.3%	8.81B	16 812	32.3
LightMTP-single	1.39B	734.5M	654.3M	4.7M	+0.3%	8.81B	16 812	25.6
LightMTP-concat	1.40B	741.6M	654.3M	11.8M	+0.9%	8.90B	16 974	26.2

Table 6: Scaling and non-scaling (token and value embeddings) parameter counts for models with 12, 18, and 24 transformer layers. Training tokens and optimizer steps for each architecture. "MTP head" is the auxiliary head(s) each variant adds on top of the NTP model. "Δ params" denotes the percentage of extra parameters the method adds compared to just NTP. Token and step budgets follow nanochat's `speedrun.sh`: a data:parameter ratio of 12 over $n_{\text{matrix}}+n_{\text{head}}$ (transformer matrices plus output head, excluding token and value embeddings). Because the ratio is applied to a parameter count that grows with the MTP head size, variants that add more parameters are trained for proportionally more steps. Future Summaries requires an already right-to-left-trained, NTP-based helper model to generate informative latent targets. "H100 GPU-h" is the approximate pretraining cost in GPU-hours on one NVIDIA H100.

final hidden state, rather than having a transformer layer of its own. This keeps the next-token path identical to that of NTP and of every other model we compare against, so the next-token capacity

is the same across the table and any difference we measure is attributable to the auxiliary objective rather than to extra depth on the next-token path.

Meta-L. The $k-1$ auxiliary heads are untied *linear* LM heads, each an independent copy of the unembedding matrix, reading the same final hidden state. This variant is also presented in Gloeckle et al. (2024), in the alternative-architecture ablation of their appendix, where heads that are a single linear layer without nonlinearity are compared against the transformer-layer heads of the main architecture. The authors report no significant improvement from the alternatives. We keep it as a separate baseline because it isolates the effect of separate unembedding matrices.

DeepSeek-v3-style MTP. DeepSeek-V3 (DeepSeek-AI et al., 2025) augments next-token prediction with $D = k - 1$ sequential MTP modules that preserve a causal chain across the predicted positions. Each module applies RMSNorm to the previous depth's representation $\boldsymbol{h}_i^{k-1}$ and to the embedding of token t_{i+k}, concatenates the two, projects the result with $M_k \in \mathbb{R}^{d\times 2d}$, and passes it through its own Transformer block. The embedding table and the output head are shared with the main model, so each module adds only M_k and one block. The per-depth cross-entropy losses are averaged and added to the NTP loss with weight λ. We keep this structure and adapt two settings to our comparison. First, we use $D = k - 1 = 3$ modules rather than the $D = 1$ of DeepSeek-V3, so that the objective supervises the same four positions as the other methods. Second, we set $\lambda = 1$, as for every other method, rather than DeepSeek-V3's schedule of 0.3 decaying to 0.1, since we do not tune loss weights per method. Each module's transformer block is randomly initialized and inherits the local attention window of our backbone's first layer (1,024 tokens) rather than attending over the full context. As in the original, the modules are discarded at inference. At depth 24 they add 99M parameters (Table 6).

Future Summaries. Future Summary Prediction (Mahajan et al., 2026) trains an auxiliary head to predict a compact representation of the entire future continuation rather than individual future tokens. We reproduce their learned-summary variant: a separate reverse language model is pretrained right-to-left on the same corpus, and the left-to-right model is then trained to predict that model's hidden state from its own. The reverse model is frozen throughout and is never used at inference. As in the original, we treat its pretraining as a one-time overhead and mostly exclude it from the token budgets in Table 6. It has the same depth, width and vocabulary as the model it supervises, so it roughly doubles the total pretraining compute of this baseline. We deviate in three aspects. First, we implement only the learned-summary variant, not the handcrafted bag-of-words summaries the original also proposes. Second, the auxiliary head is the same two-layer MLP with a GELU nonlinearity that LightMTP uses, rather than the transformer block of the original, so that the two latent methods are compared at equal head capacity. Third, we use a cosine loss on unit-normalized vectors rather than the original's ℓ_2 loss, again matching LightMTP. Targets are taken from the reverse model's final layer and predicted from the final layer of the left-to-right trunk. At depth 24 the head adds 4.7M parameters, the same as LightMTP single-target.

B Further Experiment Details

B.1 Future token anticipation

Linear probes are trained on hidden states extracted from FineWeb-Edu shard 435, which is neither in the training nor in the validation split of any model we report. Probe positions whose target token is a special token are discarded, matching the masking used for the pretraining bits-per-byte metric. For each horizon k, positions are subsampled to at most 2M and then split 80/20 into probe-train and probe-test sets, using a fixed permutation seed (42) shared across all models, so every model is probed on exactly the same positions.

B.2 LM Benchmarks

CORE and LAMBADA are evaluated from the DCLM-derived evaluation bundle shipped with `nanochat`, using its per-task few-shot counts and prompt templates unchanged. Few-shot examples are drawn so all models see identical prompts. MMLU (Hendrycks et al., 2021) is evaluated as a 10-shot multiple-choice task over all 14,042 test questions from 57 categories. Each question comes

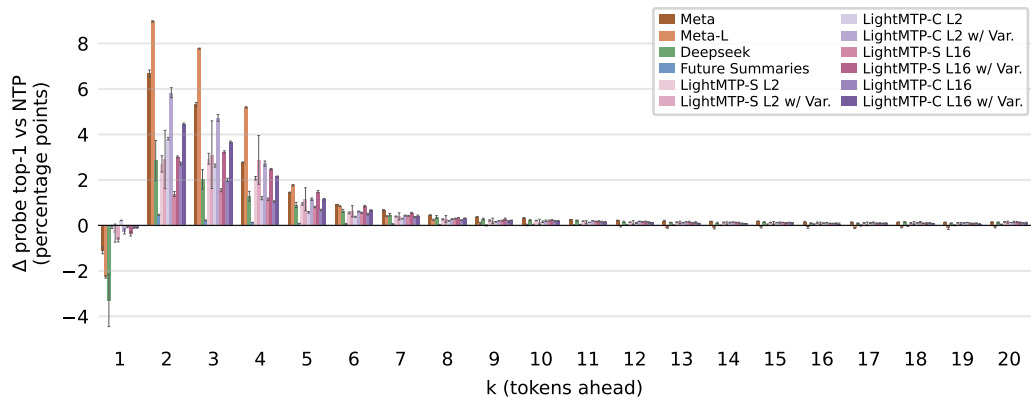


Figure 6: Probing top-1 accuracy on the models' frozen last hidden states over three seeds for $k = 1, \ldots, 20$. All MTP models exhibit higher future token anticipation than NTP, but this often comes with a performance trade-off at $k = 1$. While for $2 \leq k \leq 5$, Meta's MTP-L exhibits the strongest anticipation effect, its anticipation then gradually anneals to the other MTP models' performance until it dips *below* NTP for $k \geq 12$. LightMTP models anticipation effect for early k is moderate but it benefits from a mitigated degradation at $k = 1$.

with its four options already inserted into the prompt. We score by comparing the model's likelihood of each answer letter. Due to the models' limited pretraining budget and modest sizes, their MMLU results are not distinguishable from chance level (25%) and therefore only reported for completeness's sake in 9 in the Appendix.

B.3 Countdown

Countdown problems are generated following Teoh et al. (2026): We reserve 10% of the *targets* for out-of-distribution evaluation, so held-out problems have targets the model never saw during training. We generate 500,000 training problems, 10,000 in-distribution, and 10,000 held-out-target evaluation problems, each with four input numbers and a three-equation solution. The in-distribution evaluation set is deduplicated against the training set by exact string match, the held-out-target set is distinct by construction. Every generated solution is verified to be arithmetically valid and to reach its target. Countdown uses its own tokenizer, in which each integer below 10,000 is a single token, and eight pause tokens (Goyal et al., 2024) separate the problem from the solution.

B.4 GSM8K

For GSM8K, we evaluate on the 1,319-example test split, with eight in-context examples sampled per test item from the official 7,473-example training split. Exemplars are rendered as `Q: {question}\nA: {rationale} The answer is {answer}.`, and examples are gradually dropped if a prompt exceeds the context window.

B.5 MBPP

MBPP uses the official task-ID splits: tasks 601–974 for finetuning, tasks 511–600 for evaluation, and tasks 2–4 from the reserved prompt split as few-shot exemplars, so the exemplars never overlap the evaluation set. Prompts follow the original MBPP format (problem statement, the three assert statements, and a `[BEGIN]/[DONE]` delimited solution). Generated code is cut at `[DONE]` and executed in a sandbox against the problem's asserts, with a five-second timeout. During finetuning, the loss is masked over the prompt and applied only to the solution.

B.6 MBPP: Qualitative Examples

We provide two qualitative examples illustrating common structural failures NTP is more prone to than LightMTP.

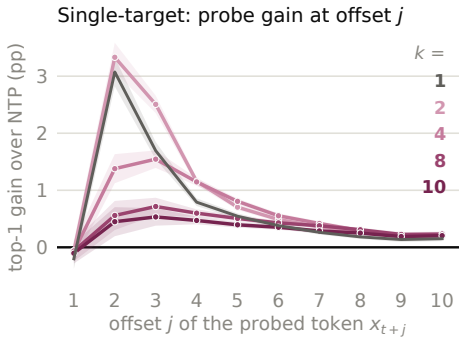
	NTP	Meta	Meta-L	Future Summaries	LightMTP-C	LightMTP-S
Over all completions						
parses as Python	96.00±1.11	**96.44**±1.16	96.07±1.01	95.33±1.49	96.41±1.43	95.70±0.17
solved	4.07±0.56	4.48±1.12	**6.37**±0.17	5.44±0.97	5.70±0.53	4.00±0.29
Structural failures (% of failing completions)						
NameError	8.27±3.66	7.93±2.37	7.16±0.44	**5.21**±0.37	6.36±1.11	5.90±1.41
SyntaxError	4.13±0.99	3.68±0.59	3.72±0.65	4.61±1.34	**3.66**±1.29	4.20±1.53
IndentationError	0.93±0.91	**0.51**±0.30	0.71±0.36	0.63±0.64	0.55±0.67	0.81±0.81
ModuleNotFoundError	0.43±0.24	0.39±0.57	0.83±0.31	0.08±0.07	0.31±0.55	**0.00**±0.00
Structural (total)	*13.76*±5.51	*12.51*±2.66	*12.42*±1.11	***10.53***±1.83	*10.88*±2.54	*10.92*±2.35
Non-structural, for contrast (% of failing)						
AssertionError (ran, wrong answer)	**58.84**±1.83	62.01±3.06	63.21±0.96	61.66±1.38	59.74±5.43	59.91±2.22
TypeError	17.83±1.70	16.49±3.96	**13.93**±1.34	17.42±2.07	20.58±1.93	20.18±0.96
IndexError	4.32±1.63	3.95±0.58	4.31±0.39	5.06±1.38	**3.18**±0.83	3.82±0.72
AttributeError	2.55±0.52	2.21±0.80	2.33±0.91	2.86±0.36	2.83±1.12	**2.16**±1.36
Timeout	0.73±0.34	1.12±0.23	1.07±0.85	**0.51**±0.07	0.75±0.36	0.73±0.52

Table 7: MBPP failure-mode breakdown, on the MBPP validation split (tasks 511–600): 90 problems $\times$ 10 samples $=$ 900 completions per model/seed, temperature 0.7, mean $\pm$ std over seeds 42, 7, and 1337. *Parses* and *solved* are percentages of *all* completions. Each failure row reports a percentage of that model's *failing* completions, re-executed against the problem's asserts and labeled by the exception raised. *Structural* sums the four exception types attributable to local naming and syntax consistency rather than task or data understanding.

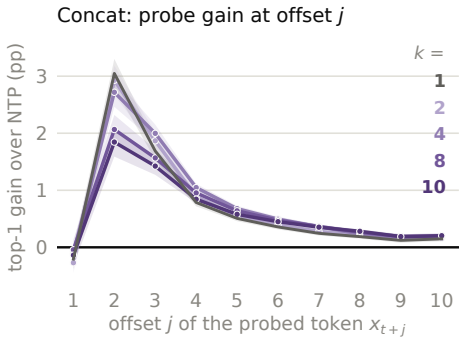
```
def find_Min_Sum(n):
    result = 0
    for i in range(2,int(math.sqrt(n)) + 1):  # uses math.sqrt, never
    imported "math"
        ...

def modular_sum(nums, m, divisor):
    if (dividor % m == 0):   # defined "divisor", referenced "dividor" a
    few tokens later
```

B.7 What MTP horizon works best?

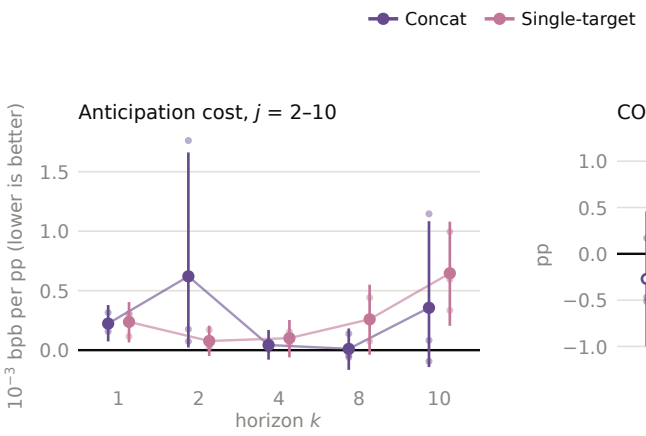
For both LightMTP variants, we run a sweep over the MTP horizon $k \in \{1, 2, 4, 8, 10\}$. Future token anticipation changes and differences in CORE results are visualized in the panels of Figure 7.



(a) Single-target: linear-probe top-1 gain over NTP for decoding x_{t+j} from the final-layer state at position t. Results over three seeds. Shade encodes k Bands: 95% CI.



(b) Concat: as in 7a, for the concatenated next-k targets.



(c) Anticipation cost per k computed over $t{+}2..t{+}10$; lower is better. Dots: per-seed costs; bars: 95% CI.

(d) CORE average over 22 tasks, difference from NTP. Dots: per-seed differences. Bars: 95% CI.

Figure 7: Effect of the prediction horizon k for LightMTP at target layer 16 (head layer 23), relative to NTP. NTP uses 5 seeds and every other configuration 3. All sweep runs use target layer 16 without VarReg ($k{=}4$ is the LightMTP L16 configuration).

B.7.1 FULL HORIZON SWEEP RESULTS

	World Knowledge						Commonsense							Passage Comprehension				Symbolic reasoning									
Method	jeopardy	bb-qa-wikidata	arc-easy	arc-challenge	openbook-qa	bb-language-identification	copa	commonsense-qa	piqa	hellaswag	hellaswag-zeroshot	winogrand	winogrande	squad	coqa	boolq	lambada-openai	bb-dyck-languages	bb-cs-algorithms	bb-repeat-copy-logic	bb-operators	lsat-ar	CORE avg	MMLU (ICL)	GSM8K (p@5)	GSM8K (strict)	Lambada (PPL)
Baseline																											
NTP	18.61 ±1.36	51.60 ±0.79	66.44 ±1.13	37.30 ±0.88	37.96 ±1.01	25.42 ±0.42	66.80 ±1.10	28.70 ±6.61	**71.58** ±0.28	51.27 ±0.36	51.11 ±0.31	69.60 ±1.55	56.76 ±0.97	35.89 ±1.26	25.79 ±1.10	52.97 ±6.59	43.49 ±0.87	10.86 ±0.96	40.44 ±3.65	3.12 ±0.00	17.81 ±1.92	25.91 ±2.45	40.43 ±0.68	24.97 ±0.49	41.97 ±2.29	4.94 ±1.44	16.00 ±0.72
Concat-targets (target layer 16)																											
Concat k=1	17.89 ±0.80	52.86 ±0.53	**67.21** ±0.53	37.63 ±0.15	**39.87** ±1.33	25.18 ±0.56	67.67 ±2.31	28.31 ±0.74	71.11 ±0.29	51.29 ±0.39	50.85 ±0.32	67.77 ±0.73	55.80 ±1.10	35.12 ±1.11	26.59 ±1.18	46.97 ±4.96	43.11 ±0.57	11.40 ±2.05	39.80 ±2.77	2.08 ±3.61	17.14 ±0.95	27.83 ±1.15	40.16 ±0.38	24.37 ±0.55	42.53 ±1.91	4.60 ±0.24	16.60 ±0.09
Concat k=2	16.97 ±2.10	51.90 ±0.59	65.80 ±0.49	36.58 ±0.64	38.60 ±0.20	25.19 ±0.74	67.33 ±2.89	23.31 ±3.44	70.62 ±1.23	50.52 ±1.37	50.11 ±1.24	67.28 ±2.44	56.35 ±1.16	35.46 ±1.67	25.98 ±0.18	51.06 ±4.71	43.21 ±0.52	10.30 ±2.76	**41.39** ±3.43	5.21 ±1.80	17.62 ±1.26	27.10 ±2.39	39.90 ±0.45	24.41 ±0.19	42.66 ±2.08	5.10 ±1.51	16.92 ±0.81
Concat k=4	18.88 ±1.20	**53.49** ±0.43	65.54 ±0.81	37.09 ±0.73	38.53 ±0.81	25.63 ±0.36	**70.00** ±3.46	25.28 ±3.42	70.64 ±0.08	51.30 ±0.11	**51.21** ±0.11	**69.84** ±0.92	57.06 ±1.47	35.49 ±1.12	26.41 ±0.48	52.59 ±0.42	43.24 ±0.44	10.43 ±4.01	37.63 ±0.99	1.04 ±1.80	**19.68** ±1.67	25.65 ±4.52	40.30 ±0.23	24.44 ±1.02	43.62 ±1.51	4.98 ±0.91	16.35 ±0.20
Concat k=8	19.00 ±1.70	51.72 ±0.14	66.39 ±0.53	**38.28** ±0.60	39.40 ±2.27	25.68 ±0.38	67.67 ±0.58	27.41 ±3.07	70.82 ±0.41	51.44 ±0.11	50.96 ±0.25	69.23 ±0.63	**58.46** ±1.02	35.94 ±1.66	26.61 ±0.49	56.32 ±3.21	**44.07** ±0.58	10.60 ±3.18	39.14 ±3.86	4.17 ±4.77	18.73 ±1.20	28.26 ±1.90	**40.92** ±0.35	**25.42** ±0.57	43.37 ±1.17	4.25 ±0.91	**15.66** ±0.27
Concat k=10	18.55 ±2.06	52.26 ±0.44	66.40 ±0.63	37.09 ±0.27	37.47 ±1.17	**25.72** ±0.65	66.00 ±1.73	**29.32** ±3.56	70.95 ±1.04	51.31 ±0.93	51.01 ±0.79	67.52 ±1.29	56.80 ±0.51	**36.71** ±0.16	**27.05** ±0.34	53.04 ±6.59	44.04 ±1.37	10.07 ±0.78	40.45 ±3.51	4.17 ±3.61	18.89 ±1.20	27.83 ±2.26	40.57 ±0.34	24.76 ±0.18	**43.97** ±0.72	4.47 ±0.99	15.97 ±1.04
Single-target (target layer 16)																											
Single-target k=1	18.58 ±2.33	52.68 ±0.67	66.68 ±0.48	37.09 ±0.40	38.40 ±0.87	25.44 ±0.50	69.33 ±2.89	27.49 ±4.96	70.91 ±0.22	51.15 ±0.21	50.71 ±0.15	69.11 ±2.02	56.93 ±0.41	36.19 ±0.83	26.93 ±1.40	51.16 ±3.32	43.28 ±0.39	11.27 ±2.50	38.23 ±1.03	0.00 ±0.00	18.25 ±1.45	28.41 ±2.62	40.37 ±0.64	24.92 ±0.38	42.81 ±0.22	5.61 ±0.57	16.60 ±0.51
Single-target k=2	18.56 ±2.06	52.31 ±0.55	66.26 ±0.43	37.94 ±0.47	38.67 ±1.89	25.39 ±0.28	66.33 ±1.15	24.46 ±6.32	70.93 ±0.32	**51.45** ±0.10	51.05 ±0.12	68.50 ±1.94	56.30 ±1.16	36.04 ±1.96	25.97 ±0.55	**59.84** ±0.85	43.71 ±0.92	11.17 ±1.07	40.28 ±2.35	2.08 ±1.80	17.30 ±3.64	**29.71** ±2.05	40.65 ±0.42	25.30 ±1.05	43.44 ±1.72	5.36 ±0.55	16.08 ±0.87
Single-target k=4	18.04 ±1.37	51.83 ±1.61	66.72 ±0.99	36.04 ±1.06	37.60 ±0.53	25.51 ±0.66	66.67 ±3.51	27.87 ±4.76	70.73 ±0.36	51.15 ±0.30	51.02 ±0.09	68.86 ±2.56	56.85 ±0.84	35.35 ±1.12	25.94 ±1.08	49.26 ±3.83	43.32 ±0.63	**11.43** ±0.95	39.44 ±2.18	4.17 ±1.80	17.46 ±2.91	27.54 ±4.04	40.13 ±0.67	24.50 ±1.05	43.82 ±1.33	**5.69** ±1.09	16.28 ±0.48
Single-target k=8	18.53 ±2.13	52.21 ±0.79	66.32 ±0.68	37.51 ±1.01	39.13 ±1.62	25.44 ±0.18	69.33 ±0.58	26.62 ±1.10	71.15 ±0.47	51.34 ±0.26	50.85 ±0.11	67.40 ±1.27	56.30 ±0.53	34.78 ±2.27	26.16 ±0.68	54.93 ±2.85	43.27 ±0.29	9.30 ±2.19	40.18 ±3.31	**6.25** ±3.12	19.05 ±0.48	26.23 ±4.29	40.56 ±0.29	25.05 ±1.12	43.77 ±2.71	4.65 ±0.79	16.10 ±0.39
Single-target k=10	**19.90** ±0.57	52.73 ±0.66	65.73 ±1.06	37.12 ±0.90	38.20 ±0.00	25.50 ±0.43	66.33 ±1.53	22.19 ±3.04	71.27 ±0.25	51.38 ±0.25	51.07 ±0.28	66.91 ±0.92	56.04 ±0.21	36.36 ±1.31	26.30 ±0.54	53.65 ±0.96	43.97 ±0.20	9.27 ±2.91	39.62 ±3.37	3.12 ±0.00	18.57 ±2.65	28.84 ±1.65	40.19 ±0.34	25.26 ±0.16	42.56 ±2.75	4.09 ±0.93	15.92 ±0.46

Table 8: depth-24 horizon sweep with target layer 16: latent MTP with concatenated next-k hidden-state targets (Concat) or the single hidden state at $t{+}k$ (Single-target), for $k \in \{1, 2, 4, 8, 10\}$, against NTP. NTP is averaged over five seeds, as in the main results tables; every other row is three seeds. At k=1, the two objectives coincide. Full CORE suite: per-task in-context accuracy (%), grouped into closed-book knowledge, everyday commonsense, passage comprehension, and symbolic reasoning, followed by the CORE average over its 22 tasks, and MMLU and GSM8K pass@5, which are reported outside the CORE average. Cells color-coded per column, green (improvement) to red (regression) vs NTP; in each column the best value is **bolded** and the runner-up underlined. CORE (centered) averages each of the 22 CORE tasks as $(\text{acc} - b)/(1 - b)$ with respect to random baseline b.

B.8 FULL LM BENCHMARK RESULTS

	World Knowledge						Commonsense							Passage Comprehension				Symbolic reasoning										
Method	**jeopardy**	**bb-qa-wikidata**	**arc-easy**	**arc-challenge**	**openbook-qa**	**bb-language-identification**	**copa**	**commonsense-qa**	**piqa**	**hellaswag**	**hellaswag-zeroshot**	**winograd**	**winogrande**	**squad**	**coqa**	**boolq**	**lambada-openai**	**bb-dyck-languages**	**bb-cs-algorithms**	**bb-repeat-copy-logic**	**bb-operators**	**lsat-ar**	**CORE avg**	**MMLU (ICL)**	**GSM8K (p@5)**	**GSM8K (strict)**	**Lambada (PPL)**	**Lambada (acc)**
Baselines																												
NTP	18.61 ±1.36	51.60 ±0.79	66.44 ±1.13	**37.30** ±0.88	37.96 ±1.01	25.42 ±0.42	66.80 ±1.10	28.70 ±6.61	**71.58** ±0.28	51.27 ±0.36	51.11 ±0.31	69.60 ±1.55	56.76 ±0.97	35.89 ±1.26	25.79 ±1.10	52.97 ±6.59	43.49 ±0.87	10.86 ±0.96	40.44 ±3.65	3.12 ±0.00	17.81 ±1.92	25.91 ±2.45	40.43 ±0.68	24.97 ±0.49	41.97 ±2.29	4.94 ±1.44	16.00 ±0.72	39.00 ±0.68
Meta	14.49 ±1.62	49.84 ±0.89	64.49 ±0.36	34.95 ±0.49	37.67 ±0.92	24.88 ±0.25	68.00 ±1.00	22.88 ±2.74	70.18 ±0.36	48.50 ±0.72	48.24 ±0.60	66.06 ±0.76	56.01 ±1.34	35.19 ±1.68	26.13 ±1.00	59.06 ±3.60	42.12 ±0.86	12.27 ±1.01	39.32 ±1.51	3.12 ±3.12	18.25 ±0.99	23.19 ±3.29	39.31 ±0.69	24.65 ±0.97	41.45 ±1.43	4.09 ±0.08	18.66 ±0.84	38.00 ±0.68
Meta-L	18.76 ±0.95	50.69 ±0.58	65.76 ±0.29	**37.30** ±0.83	38.48 ±0.86	24.89 ±0.26	67.20 ±2.17	26.22 ±2.31	70.95 ±0.58	51.02 ±0.30	50.79 ±0.29	67.69 ±1.11	56.56 ±1.67	**38.53** ±0.68	**28.16** ±0.37	45.22 ±6.03	42.80 ±0.68	11.38 ±0.82	37.14 ±2.95	5.62 ±1.40	18.00 ±1.23	24.43 ±2.77	39.89 ±0.42	24.54 ±0.25	41.91 ±0.88	4.82 ±1.19	18.13 ±0.48	38.00 ±0.68
DeepSeek MTP	6.15 ±4.58	43.79 ±5.14	59.78 ±3.76	30.49 ±2.62	33.76 ±2.04	25.03 ±0.39	63.20 ±2.77	28.68 ±4.94	67.52 ±1.43	41.48 ±4.25	41.73 ±4.21	62.34 ±3.45	52.64 ±1.78	21.98 ±1.46	19.71 ±4.66	57.86 ±3.32	35.77 ±4.33	11.62 ±2.64	39.12 ±1.55	1.25 ±1.71	14.57 ±3.98	**29.30** ±1.52	35.81 ±2.60	25.14 ±0.48	39.17 ±2.11	3.85 ±0.68	31.49 ±11.42	34.00 ±0.66
Future Summaries	17.95 ±1.61	51.71 ±1.39	65.60 ±0.47	36.80 ±0.57	38.00 ±0.87	25.41 ±0.31	67.00 ±2.00	26.67 ±3.14	70.67 ±0.20	51.16 **51.23** ±0.19	±0.23	69.35 ±2.02	**57.56** ±0.73	35.82 ±0.48	26.49 ±1.02	52.53 ±3.42	**44.35** ±0.45	10.00 ±3.90	38.56 ±1.29	2.08 ±1.80	17.78 ±1.53	28.84 ±2.62	40.25 ±0.37	25.28 ±0.49	41.19 ±0.53	4.95 ±1.25	**15.57** ±0.33	**40.00** ±0.68
LightMTP: concat-targets																												
concat-targets L16	18.88 ±1.20	**53.49** ±0.43	65.54 ±0.81	37.09 ±0.73	38.53 ±0.81	25.63 ±0.36	**70.00** ±1.46	25.28 ±3.42	70.64 ±0.08	51.30 ±0.11	51.21 ±0.11	**69.84** ±0.92	57.06 ±1.47	35.49 ±1.12	26.41 ±0.48	52.59 ±0.42	43.24 ±0.44	10.43 ±4.01	37.63 ±0.99	1.04 ±1.80	**19.68** ±1.67	25.65 ±4.52	40.30 ±0.23	24.44 ±1.02	43.62 ±1.51	4.98 ±0.91	16.35 ±0.20	38.00 ±0.68
concat-targets L16 +VarReg	16.47 ±2.68	52.21 ±0.31	65.64 ±0.76	35.98 ±0.57	38.47 ±0.76	25.26 ±0.25	66.67 ±2.08	29.46 ±0.90	70.33 ±0.60	50.38 ±0.37	50.35 ±0.20	68.62 ±0.56	56.46 ±0.46	34.85 ±1.12	25.96 ±0.81	55.95 ±1.89	42.63 ±0.57	11.03 ±1.05	40.78 ±4.30	3.12 ±0.00	18.73 ±2.44	24.64 ±4.04	40.18 ±0.43	23.96 ±0.35	43.42 ±2.97	5.69 ±0.33	16.84 ±0.41	38.00 ±0.68
concat-targets L2	19.04 ±1.83	52.49 ±0.24	**66.79** ±0.34	37.20 ±1.40	38.53 ±0.95	25.53 ±0.15	66.00 ±1.00	29.02 ±3.58	71.07 ±0.95	51.17 ±0.26	51.04 ±0.21	68.25 ±0.42	56.22 ±0.79	35.64 ±1.13	25.68 ±0.30	55.75 ±2.05	43.28 ±0.68	12.00 ±1.85	40.05 ±0.74	2.08 ±1.80	18.25 ±0.27	28.26 ±1.90	**40.61** ±0.33	24.79 ±0.80	41.65 ±2.33	4.98 ±0.89	16.16 ±0.37	38.00 ±0.68
concat-targets L2 +VarReg	15.70 ±1.02	50.35 ±0.55	65.24 ±0.61	35.72 ±0.10	38.40 ±1.25	25.50 ±0.29	66.67 ±2.52	23.61 ±1.79	69.84 ±0.54	49.59 ±0.44	49.48 ±0.24	67.16 ±1.69	56.35 ±1.06	33.94 ±0.65	25.90 ±0.54	58.73 ±5.04	43.29 ±0.95	10.00 ±3.12	38.94 ±1.73	2.08 ±3.61	18.57 ±3.81	28.26 ±1.74	39.70 ±0.33	25.02 ±0.17	40.79 ±1.76	4.80 ±1.23	17.08 ±0.76	38.00 ±0.68
concat-targets L16 (k=10)	18.55 ±2.06	52.26 ±0.44	66.40 ±0.63	37.09 ±0.27	37.47 ±1.17	**25.72** ±0.65	66.00 ±1.73	29.32 ±3.56	70.95 ±1.04	51.31 ±0.93	51.01 ±0.79	67.52 ±1.29	56.80 ±0.51	36.71 ±0.16	27.05 ±0.34	53.04 ±6.59	44.04 ±1.37	10.07 ±0.78	40.45 ±3.51	4.17 ±3.61	18.89 ±1.20	27.83 ±2.26	40.57 ±0.34	24.76 ±0.18	**43.97** ±0.72	4.47 ±0.99	15.97 ±1.04	38.00 ±0.68
LightMTP: single-target																												
single-target L16	18.04 ±1.37	51.83 ±1.61	66.72 ±0.99	36.04 ±1.06	37.60 ±0.53	25.51 ±0.66	66.67 ±3.51	27.87 ±4.76	70.73 ±0.36	51.15 ±0.30	51.02 ±0.09	68.86 ±2.56	56.85 ±0.84	35.35 ±1.12	25.94 ±1.08	49.26 ±3.83	43.32 ±0.63	11.43 ±0.95	39.44 ±2.18	4.17 ±1.80	17.46 ±2.91	27.54 ±4.04	40.13 ±0.67	24.50 ±1.05	43.82 ±1.33	5.69 ±1.09	16.28 ±0.48	38.00 ±0.68
single-target L16 +VarReg	17.56 ±1.11	51.96 ±1.38	65.60 ±0.34	35.24 ±0.73	36.80 ±1.59	25.51 ±0.30	65.33 ±1.18	**31.59** ±3.26	70.13 ±0.96	50.23 ±0.29	50.16 ±0.55	68.25 ±0.21	56.62 ±1.60	33.94 ±0.63	25.86 ±0.60	**60.73** ±2.10	42.38 ±0.25	10.17 ±1.50	41.09 ±2.85	2.08 ±1.80	18.10 ±1.65	27.10 ±1.96	40.29 ±0.46	25.01 ±0.31	41.39 ±1.14	4.75 ±0.64	17.53 ±0.35	38.00 ±0.68
single-target L2	17.46 ±1.70	51.05 ±1.87	65.07 ±2.01	35.72 ±1.25	38.00 ±0.60	25.69 ±0.44	67.67 ±2.08	30.00 ±2.02	70.48 ±1.20	50.17 ±1.71	49.98 ±1.63	67.03 ±1.83	55.56 ±1.66	35.03 ±0.37	25.97 ±1.23	57.99 ±3.40	42.70 ±0.93	**12.93** ±1.46	**41.31** ±0.57	**6.25** ±0.00	16.19 ±0.48	27.39 ±1.90	40.44 ±0.63	25.14 ±0.72	42.23 ±2.48	**5.94** ±0.58	17.00 ±0.94	38.00 ±0.68
single-target L2 +VarReg	15.43 ±1.95	51.16 ±1.40	64.84 ±0.18	35.21 ±0.66	36.60 ±1.59	25.44 ±0.35	66.00 ±1.73	23.89 ±4.94	69.89 ±0.71	49.17 ±0.33	49.05 ±0.55	66.79 ±0.92	54.46 ±0.85	34.09 ±2.77	25.35 ±1.41	57.17 ±3.18	42.45 ±0.15	10.47 ±1.80	37.27 ±3.12	2.08 ±3.61	17.30 ±1.67	25.80 ±4.09	39.09 ±0.42	24.45 ±0.41	42.13 ±0.42	3.61 ±0.23	17.68 ±0.33	38.00 ±0.68
single-target L16 (k=10)	**19.90** ±0.57	52.73 ±0.66	65.73 ±1.06	37.12 ±0.90	38.20 ±0.00	25.50 ±0.43	66.33 ±1.53	22.19 ±3.04	71.27 ±0.25	**51.38** ±0.25	51.07 ±0.28	66.91 ±0.92	56.04 ±0.21	36.36 ±1.31	26.30 ±0.54	53.65 ±0.96	43.97 ±0.20	9.27 ±2.91	39.62 ±3.37	3.12 ±0.00	18.57 ±2.65	28.84 ±1.65	40.19 ±0.34	25.26 ±0.16	42.56 ±2.75	4.09 ±0.93	15.92 ±0.46	38.00 ±0.68
single-target L2 (k=10)	17.02 ±1.21	50.77 ±2.98	65.32 ±0.44	35.92 ±1.51	**39.20** ±0.92	25.61 ±0.64	67.33 ±1.53	23.15 ±1.61	70.53 ±1.72	50.40 ±2.19	49.92 ±2.04	67.64 ±1.12	55.72 ±2.26	36.80 ±1.19	25.77 ±1.22	54.33 ±3.75	43.08 ±0.87	12.43 ±0.57	38.94 ±2.41	4.17 ±3.61	17.78 ±0.99	28.12 ±2.62	40.00 ±0.84	**25.73** ±0.68	41.82 ±3.14	5.36 ±0.29	16.72 ±1.05	38.00 ±0.68

Table 9: Full depth-24 language modeling benchmark results: full CORE benchmark suite (per-task ICL accuracy), CORE average, MMLU (ICL), GSM8K pass@5 under strict and relaxed matchers and Lambada perplexity and accuracy; mean ± std over three seeds for every row, five for NTP. Every MTP model uses k=4 targets. Every run is trained with parameter-matched budget, so the number of steps follows each model's parameter count. Meta-L trains on 20.7%, DeepSeek MTP on 13.6%, and Meta on 11.6% more tokens than NTP. CORE (centered) averages each of the 22 CORE tasks as $(\text{acc} - b)/(1 - b)$ with respect to random baseline b. **LAMBADA columns.** The base model (not a Lambada-finetuned variant) on the LAMBADA test split: target-word perplexity (lower is better) and accuracy.

C FULL LM BENCHMARK RESULTS FOR OTHER MODEL SIZES

C.1 DEPTH-12 MODELS

Method	World Knowledge						Commonsense							Passage Comprehension				Symbolic reasoning										
	jeopardy	bb-qa-wikidata	arc-easy	arc-challenge	openbook-qa	bb-language-identification	copa	commonsense-qa	piqa	hellaswag	hellaswag-zeroshot	winograd	winogrande	squad	coqa	boolq	lambada-openai	bb-dyck-languages	bb-cs-algorithms	bb-repeat-copy-logic	bb-operators	lsat-ar	CORE avg	MMLU (ICL)	GSM8K (p@5)	GSM8K (strict)	Lambada (PPL)
Baselines																												
NTP	0.50 ±0.43	29.25 ±1.26	50.59 ±0.81	24.43 ±0.90	29.80 ±1.93	25.09 ±0.04	58.67 ±3.06	31.31 ±1.82	62.80 ±0.28	32.11 ±0.10	32.31 ±0.06	58.00 ±1.29	51.35 ±0.20	6.13 ±1.62	11.57 ±0.20	52.52 ±3.17	27.35 ±0.66	11.53 ±0.67	41.26 ±1.99	2.08 ±1.80	10.79 ±2.62	26.81 ±1.00	30.74 ±0.09	24.50 ±0.48	38.24 ±0.77	**3.99** ±0.34	72.99 ±3.22
Meta	0.66 ±0.22	**32.82** ±1.51	50.87 ±0.96	25.51 ±0.64	30.60 ±0.35	25.29 ±0.50	**61.67** ±2.08	28.67 ±3.30	63.02 ±0.93	32.16 ±0.23	32.30 ±0.16	55.19 ±0.92	50.57 ±1.04	7.69 ±3.20	12.31 ±1.61	53.12 ±7.00	**28.76** ±0.14	11.03 ±2.04	40.56 ±0.80	0.00 ±0.00	9.37 ±1.92	26.67 ±1.53	30.86 ±0.21	24.88 ±0.44	37.25 ±0.82	3.34 ±1.45	67.56 ±0.86
Meta-L	1.04 ±0.39	31.42 ±1.39	50.76 ±0.58	**26.59** ±0.18	**31.93** ±0.95	25.16 ±0.21	**61.67** ±0.58	23.89 ±1.49	62.62 ±0.16	**32.95** ±0.24	**33.49** ±0.33	58.49 ±0.92	51.33 ±1.42	10.65 ±1.99	13.30 ±1.13	54.37 ±3.80	27.55 ±0.85	12.03 ±1.78	36.52 ±1.76	1.04 ±1.80	9.21 ±0.73	24.49 ±2.19	30.93 ±0.26	24.32 ±0.67	34.93 ±1.07	3.29 ±0.22	**66.95** ±3.27
DeepSeek MTP	**1.21** ±0.70	31.01 ±2.02	50.91 ±0.79	26.05 ±0.13	29.60 ±0.87	25.41 ±0.24	57.67 ±4.62	29.98 ±3.38	62.13 ±0.29	32.15 ±0.17	32.50 ±0.27	57.14 ±1.47	51.43 ±0.30	**10.90** ±2.16	**13.69** ±0.90	**58.64** ±1.70	27.60 ±0.36	11.10 ±0.95	40.86 ±1.52	2.08 ±1.80	11.11 ±0.73	26.67 ±2.05	**31.36** ±0.40	24.87 ±0.53	39.55 ±4.33	3.49 ±0.46	75.93 ±2.73
Future Summaries	0.28 ±0.09	28.39 ±1.02	50.35 ±0.71	25.65 ±0.21	30.87 ±0.81	25.07 ±0.32	55.67 ±2.31	31.45 ±3.74	62.37 ±0.22	32.21 ±0.06	32.50 ±0.14	57.88 ±1.68	51.09 ±0.46	7.51 ±2.29	10.94 ±0.20	51.38 ±1.78	26.59 ±0.34	9.47 ±0.80	40.63 ±1.92	**3.12** ±0.00	**12.54** ±1.37	**28.26** ±0.87	30.65 ±0.08	24.65 ±0.53	35.43 ±2.93	3.13 ±0.49	75.62 ±3.24
LightMTP: concat-targets																												
concat-targets L8	0.27 ±0.10	30.23 ±0.67	50.35 ±0.79	24.66 ±0.68	30.60 ±0.35	25.35 ±0.23	57.33 ±1.53	31.45 ±2.26	62.93 ±0.30	32.18 ±0.25	32.26 ±0.12	55.92 ±1.12	51.85 ±0.60	4.50 ±0.93	10.26 ±1.44	51.55 ±2.12	26.96 ±0.20	11.27 ±2.51	37.78 ±2.39	2.08 ±1.80	11.43 ±1.65	27.68 ±3.05	30.40 ±0.11	24.72 ±0.59	38.34 ±0.77	3.94 ±0.42	75.67 ±1.47
concat-targets L8 +VarReg	0.76 ±0.45	29.69 ±1.98	**51.37** ±0.74	25.46 ±0.79	29.93 ±0.76	25.17 ±0.16	59.67 ±1.53	27.33 ±5.33	62.89 ±0.48	32.02 ±0.34	32.31 ±0.22	57.63 ±2.70	51.28 ±1.60	6.93 ±2.20	11.83 ±1.31	54.43 ±3.76	27.23 ±0.36	11.03 ±0.64	41.11 ±1.63	**3.12** ±0.00	8.89 ±0.55	26.81 ±1.65	30.77 ±0.50	24.57 ±0.12	39.12 ±0.72	3.97 ±0.70	75.87 ±4.06
concat-targets L2	0.28 ±0.08	30.21 ±3.06	50.49 ±1.18	25.26 ±1.40	31.13 ±1.33	**25.64** ±0.14	60.67 ±2.08	26.18 ±1.92	62.57 ±0.38	32.29 ±0.08	32.23 ±0.12	56.53 ±2.60	49.49 ±0.83	8.55 ±1.73	11.12 ±0.88	54.75 ±1.87	26.57 ±0.45	9.87 ±1.51	38.99 ±2.35	2.08 ±1.80	9.21 ±0.27	26.23 ±2.76	30.47 ±0.20	24.60 ±0.11	35.66 ±2.85	3.49 ±0.15	74.31 ±2.22
concat-targets L2 +VarReg	0.49 ±0.12	26.93 ±1.87	50.24 ±0.60	25.11 ±0.64	31.53 ±0.95	25.40 ±0.12	55.67 ±3.79	32.57 ±2.99	62.51 ±0.30	31.34 ±0.70	31.46 ±0.58	**59.83** ±0.56	51.12 ±0.30	7.54 ±1.22	11.27 ±1.41	51.36 ±3.49	25.93 ±1.07	10.40 ±2.55	41.16 ±0.58	2.08 ±1.80	9.05 ±0.48	26.38 ±2.96	30.43 ±0.59	24.90 ±0.80	35.61 ±1.89	3.13 ±1.08	83.83 ±8.34
LightMTP: single-target																												
single-target L8	0.57 ±0.19	29.37 ±0.73	50.01 ±0.62	26.08 ±0.27	30.53 ±1.80	25.17 ±0.14	58.67 ±2.31	31.34 ±2.40	63.00 ±0.22	32.09 ±0.28	32.26 ±0.21	56.65 ±1.29	50.64 ±0.90	7.70 ±2.83	12.71 ±0.71	52.60 ±2.84	26.62 ±0.65	11.20 ±2.03	40.10 ±1.98	**3.12** ±0.00	10.63 ±1.98	26.96 ±0.75	30.82 ±0.36	24.85 ±0.57	38.49 ±1.00	2.78 ±0.86	76.23 ±2.74
single-target L8 +VarReg	0.49 ±0.24	31.59 ±2.89	50.70 ±0.32	25.88 ±0.42	30.60 ±0.92	25.55 ±0.25	58.00 ±3.00	24.68 ±1.51	62.22 ±0.69	31.95 ±0.25	32.16 ±0.12	57.88 ±2.77	50.88 ±0.18	6.73 ±1.41	11.51 ±0.52	49.71 ±7.17	27.40 ±0.26	10.63 ±0.81	40.61 ±2.26	**3.12** ±0.00	9.37 ±0.55	25.94 ±1.81	30.35 ±0.52	24.59 ±0.62	**40.41** ±3.41	3.77 ±0.70	73.53 ±3.56
single-target L2	0.31 ±0.27	28.86 ±2.42	50.39 ±0.86	25.51 ±0.92	30.47 ±1.40	25.34 ±0.35	58.67 ±2.08	**32.84** ±3.39	**63.15** ±0.41	32.16 ±0.19	32.35 ±0.33	56.17 ±3.52	51.35 ±1.12	5.29 ±3.45	10.93 ±1.18	54.39 ±2.57	27.08 ±0.28	**13.13** ±1.42	**42.12** ±2.65	2.08 ±1.80	10.79 ±1.92	25.36 ±3.49	30.85 ±0.30	**25.14** ±0.30	34.32 ±2.09	3.54 ±0.57	72.25 ±2.43
single-target L2 +VarReg	0.22 ±0.21	26.83 ±0.62	49.75 ±0.36	25.51 ±0.60	30.00 ±0.00	25.37 ±0.22	57.33 ±0.58	31.75 ±5.86	62.30 ±0.16	31.27 ±0.20	31.63 ±0.15	56.90 ±2.02	**52.04** ±1.39	6.31 ±1.08	11.14 ±0.30	46.59 ±6.70	26.96 ±0.52	9.23 ±2.68	40.03 ±2.35	1.04 ±1.80	10.63 ±2.71	24.78 ±3.04	29.89 ±0.47	25.13 ±0.51	38.67 ±2.25	2.35 ±0.55	81.00 ±5.88

Table 10: depth-12 results: full CORE benchmark suite (per-task ICL accuracy), CORE average, MMLU (ICL), GSM8K pass@5 under strict and relaxed matchers and Lambada perplexity and accuracy; mean ± std over three seeds for every row. Every MTP model uses k=4 targets. Every run is trained with parameter-matched budget, so the number of steps follows each model's parameter count. Meta-L therefore trains on 68.6% and DeepSeek MTP on 22.5% more tokens than NTP, and Meta on 19.3%, the most likely explanation for their improvements over NTP at this size. CORE (centered) averages each of the 22 CORE tasks as $(\text{acc} - b)/(1 - b)$ with respect to random baseline b. **LAMBADA columns.** The base model (not a Lambada-finetuned variant) on the LAMBADA test split: target-word perplexity (lower is better) and accuracy.

C.2 DEPTH-18 MODELS

Method	jeopardy	bb-qa-wikidata	arc-easy	arc-challenge	openbook-qa	bb-language-identification	copa	commonsense-qa	piqa	hellaswag	hellaswag-zeroshot	winograd	winogrande	squad	coqa	boolq	lambada-openai	bb-dyck-languages	bb-cs-algorithms	bb-repeat-copy-logic	bb-operators	lsat-ar	CORE avg	MMLU (ICL)	GSM8K (p@5)	GSM8K (strict)	Lambada (PPL)
	World Knowledge						**Commonsense**							**Passage Comprehension**				**Symbolic reasoning**									
Baselines																											
NTP	4.11 ±2.27	44.14 ±1.95	59.83 ±0.53	**32.05** ±0.74	34.93 ±1.10	25.40 ±0.34	64.00 ±0.00	29.59 ±3.79	67.03 ±0.09	41.47 ±0.57	41.77 ±0.42	61.54 ±2.29	53.09 ±1.51	23.64 ±1.90	19.66 ±1.07	57.43 ±2.69	36.10 ±1.22	11.97 ±1.44	41.26 ±1.87	2.08 ±1.80	15.71 ±0.48	26.09 ±2.17	36.04 ±0.25	24.32 ±0.60	38.46 ±1.45	4.12 ±0.65	27.85 ±1.20
Meta	5.94 ±1.38	43.98 ±0.53	59.30 ±0.36	29.10 ±1.02	34.87 ±1.47	24.91 ±0.36	64.67 ±0.58	26.73 ±5.17	67.10 ±0.03	41.29 ±0.23	41.27 ±0.21	63.74 ±2.29	53.75 ±0.16	**26.53** ±0.30	**21.68** ±0.21	53.43 ±2.80	**36.57** ±1.23	12.03 ±0.45	41.69 ±2.39	1.04 ±1.80	16.35 ±0.27	24.20 ±3.75	35.92 ±0.17	24.14 ±0.65	39.55 ±3.83	4.32 ±0.66	29.26 ±2.29
Meta-L	6.69 ±0.91	43.16 ±0.71	59.64 ±0.99	31.14 ±0.68	35.33 ±1.10	25.01 ±0.52	63.00 ±3.00	23.40 ±2.53	66.85 ±0.28	**42.20** ±0.31	**42.23** ±0.51	62.27 ±1.68	**54.88** ±1.35	24.01 ±1.48	21.02 ±1.44	51.80 ±2.46	36.35 ±0.32	9.60 ±1.71	37.30 ±2.18	2.08 ±1.80	14.44 ±2.40	24.35 ±3.56	35.31 ±0.14	24.30 ±0.26	38.62 ±1.61	3.74 ±0.09	29.21 ±0.86
DeepSeek MTP	1.45 ±0.88	35.31 ±3.22	53.58 ±1.39	27.13 ±1.20	31.27 ±1.67	25.37 ±0.41	60.33 ±2.08	27.27 ±2.68	64.35 ±0.69	33.53 ±1.20	33.67 ±1.12	59.34 ±0.97	52.14 ±0.92	11.56 ±3.64	13.05 ±1.21	54.86 ±8.45	27.41 ±1.45	11.97 ±3.95	41.09 ±0.38	2.08 ±1.80	12.70 ±4.05	24.64 ±1.26	32.00 ±0.80	24.21 ±0.52	36.95 ±2.07	3.06 ±0.61	67.80 ±10.47
Future Summaries	6.85 ±0.81	44.54 ±0.78	60.03 ±0.34	31.46 ±0.85	34.80 ±0.40	25.64 ±0.30	63.33 ±2.52	26.10 ±1.44	**67.63** ±0.22	41.54 ±0.06	41.76 ±0.06	62.88 ±2.35	53.62 ±0.84	23.81 ±3.07	20.16 ±0.15	54.56 ±5.42	36.39 ±0.46	**13.53** ±1.39	39.87 ±1.15	1.04 ±1.80	16.98 ±1.20	25.51 ±5.00	36.00 ±0.27	24.88 ±0.66	40.06 ±3.38	**4.57** ±0.59	27.45 ±0.17
LightMTP: concat-targets																											
concat-targets L12	5.45 ±1.40	44.32 ±0.32	60.10 ±1.13	31.88 ±0.34	35.40 ±0.72	25.44 ±0.17	63.67 ±1.15	26.54 ±2.36	67.23 ±0.69	41.72 ±0.14	42.19 ±0.28	63.86 ±1.84	53.17 ±0.96	23.01 ±1.89	19.93 ±0.33	60.47 ±1.38	36.46 ±0.67	12.17 ±0.71	41.46 ±3.08	2.08 ±3.61	16.83 ±1.45	25.94 ±1.40	36.33 ±0.30	24.47 ±0.12	38.19 ±0.74	4.47 ±0.27	**27.35** ±0.40
concat-targets L12 +VarReg	6.19 ±1.39	46.41 ±0.49	58.98 ±0.90	31.51 ±0.42	34.67 ±0.70	25.33 ±0.43	63.00 ±0.00	28.88 ±2.27	67.41 ±0.29	41.40 ±0.29	41.73 ±0.17	**64.22** ±0.21	53.28 ±0.62	21.48 ±1.88	20.10 ±0.78	**60.98** ±1.04	35.84 ±0.48	12.90 ±1.82	40.58 ±1.25	**3.12** ±0.00	14.76 ±1.72	25.22 ±1.30	36.27 ±0.29	24.65 ±0.19	39.10 ±0.84	3.74 ±0.49	28.90 ±0.19
concat-targets L2	4.83 ±0.62	44.07 ±1.25	**60.55** ±1.08	31.34 ±0.34	**35.67** ±0.81	**25.65** ±0.18	62.67 ±1.15	25.53 ±2.29	67.28 ±0.21	41.95 ±0.21	**42.23** ±0.11	63.98 ±1.29	53.14 ±1.12	23.77 ±1.24	19.47 ±1.42	56.52 ±5.39	35.72 ±0.68	13.03 ±0.40	41.26 ±2.54	2.08 ±1.80	16.83 ±1.20	**26.38** ±2.96	36.09 ±0.38	**25.45** ±0.94	**40.59** ±2.31	4.22 ±0.78	27.66 ±0.79
concat-targets L2 +VarReg	4.08 ±1.87	43.78 ±0.33	58.75 ±0.95	29.92 ±0.50	34.07 ±0.81	25.49 ±0.11	64.33 ±4.16	24.71 ±2.79	66.50 ±0.26	40.26 ±0.17	40.71 ±0.20	61.78 ±2.15	52.14 ±1.23	21.12 ±2.27	19.97 ±0.58	56.04 ±5.26	34.78 ±0.37	11.07 ±1.97	39.04 ±2.27	1.04 ±1.80	16.03 ±1.80	25.94 ±1.76	35.07 ±0.79	25.21 ±0.64	40.36 ±1.38	4.17 ±0.08	30.51 ±1.03
LightMTP: single-target																											
single-target L12	5.09 ±1.18	44.45 ±0.18	60.04 ±0.68	31.80 ±0.73	34.73 ±0.99	25.53 ±0.40	**65.33** ±0.58	**31.61** ±2.09	67.41 ±0.52	41.80 ±0.43	41.98 ±0.49	63.98 ±1.29	52.85 ±0.41	25.16 ±0.59	20.17 ±0.10	57.95 ±2.26	35.62 ±0.63	12.20 ±0.53	**42.58** ±0.35	1.04 ±1.80	16.03 ±1.20	21.74 ±3.14	36.32 ±0.23	24.56 ±0.08	39.73 ±1.91	4.09 ±1.51	28.78 ±0.39
single-target L12 +VarReg	5.76 ±0.98	**46.69** ±0.64	59.62 ±0.47	31.29 ±1.37	34.53 ±0.99	25.24 ±0.37	64.00 ±1.00	26.45 ±3.89	67.28 ±0.22	41.59 ±0.28	41.57 ±0.22	63.25 ±0.76	53.14 ±1.18	21.73 ±1.70	20.35 ±1.58	56.91 ±2.75	35.76 ±0.37	12.37 ±1.70	41.87 ±0.74	2.08 ±1.80	14.60 ±0.27	25.94 ±2.83	36.00 ±0.48	24.77 ±0.62	37.73 ±0.44	3.46 ±0.94	29.15 ±0.26
single-target L2	**7.13** ±0.57	44.16 ±1.87	60.28 ±1.31	31.97 ±0.05	34.40 ±0.35	25.57 ±0.31	**65.33** ±1.53	25.01 ±5.99	67.19 ±0.46	41.59 ±0.54	41.90 ±0.54	61.29 ±0.56	52.59 ±2.05	25.06 ±0.84	19.88 ±1.21	60.43 ±0.52	36.05 ±1.22	11.60 ±0.26	42.37 ±1.53	**3.12** ±0.00	**18.57** ±1.72	26.23 ±2.80	**36.44** ±0.51	24.80 ±0.98	38.21 ±2.58	3.89 ±0.50	27.70 ±2.09
single-target L2 +VarReg	4.05 ±0.89	43.33 ±0.51	57.11 ±0.79	29.52 ±0.43	32.80 ±0.20	25.35 ±0.06	61.00 ±2.65	25.63 ±2.57	66.03 ±0.14	40.01 ±0.05	40.36 ±0.27	60.32 ±2.60	51.72 ±2.13	20.71 ±1.53	18.74 ±0.21	55.16 ±3.73	35.62 ±0.19	11.97 ±0.81	39.27 ±1.75	**3.12** ±0.00	16.19 ±0.48	23.91 ±1.57	34.63 ±0.38	24.78 ±0.47	38.82 ±2.97	4.17 ±0.20	30.29 ±1.11

Table 11: depth-18 results: full CORE benchmark suite (per-task ICL accuracy), CORE average, MMLU (ICL), GSM8K pass@5 under both matchers and Lambada perplexity and accuracy; mean ± std over three seeds for every row. Every MTP model uses k=4 targets; "LightMTP" layer 12 is d24's target layer 16 scaled to depth 18, layer 2 is kept as is. Every run is trained with a parameter-matched budget, so the number of steps follows each model's parameter count; Meta MTP therefore trains on 34.9% and DeepSeek MTP on 17.2% more tokens than NTP. Cells color-coded vs NTP. CORE (centered) averages each of the 22 CORE tasks as $(\text{acc} - b)/(1 - b)$ with respect to random baseline b. **Lambada columns.** The base model (not a Lambada-finetuned variant) on the LAMBADA test split: target-word perplexity (lower is better) and accuracy.