Multiple Token Divergence: Measuring and Steering In-Context Computation Density

Anonymous Author(s)
Affiliation
Address
email

Abstract

Measuring the in-context computational effort of language models is a key challenge, as metrics like next-token loss fail to capture reasoning complexity. Prior methods based on latent state compressibility can be invasive and unstable. We propose Multiple Token Divergence (MTD), a simple measure of computational effort defined as the KL divergence between a model's full output distribution and that of a shallow, auxiliary prediction head. MTD can be computed directly from pre-trained models with multiple prediction heads, requiring no additional training. Building on this, we introduce Divergence Steering, a novel decoding method to control the computational character of generated text. We empirically show that MTD is more effective than prior methods at distinguishing complex tasks from simple ones. On mathematical reasoning benchmarks, MTD correlates positively with problem difficulty. Lower MTD is associated with more accurate reasoning. MTD provides a practical, lightweight tool for analyzing and steering the computational dynamics of language models.

1 Introduction

To solve unfamiliar and challenging problems, language models must perform sophisticated in-context computation [Brown et al., 2020, Lewkowycz et al., 2022]. Can we tell whether, and to what extent, a model is making use of its computational capacity at any given moment? Next-token prediction loss offers little insight [Schmidhuber, 1991a,b], as any particular reduction in loss can, in principle, be arbitrarily difficult to achieve [Bennett, 1988]. A more promising approach is to quantify meaningful computation by measuring the entropy, or incompressibility, of a model's latent representations [Skean et al., 2025, Herrmann et al., 2025]. This concept is rooted in the minimum description length principle [Grünwald, 2007, Vitányi, 2006, Elmoznino et al., 2024]: if the most compact description of a sequence's structure, given the training data, is still long, then predicting that sequence is demanding due to a large search space. Based on this, the Prediction of Hidden States (PHi) loss was proposed as a measure of in-context computational complexity, quantifying the per-token information gain in a model's latent space [Herrmann et al., 2025]. While promising, the PHi framework introduces significant practical challenges: it requires inserting a noisy information bottleneck that can degrade model performance, needs further model training which can be unstable, and is highly sensitive to its precise placement and the weighting of multiple loss terms.

In this work, we propose a simplified and more direct measure, Multiple Token Divergence (MTD), which quantifies information gain in the model's output distribution. The core insight is simple: if a shallow computational shortcut (e.g., a single transformer block) can approximate the full model's prediction, then the model is not performing particularly complex computation. If, however, there is a significant divergence between these two predictions, we can conclude that the model is leveraging its deeper computational capacity. MTD is straightforward to implement and can even be computed directly using the Multiple Token Prediction (MTP) modules that some modern pre-trained models already possess, requiring no additional fine-tuning. In addition to this measure, we present

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Divergence Steering, a novel decoding method that uses the MTD signal to control the computational character of the generated output. We empirically demonstrate that MTD is more effective than prior methods at distinguishing complex difficult tasks from simple ones. We also discover intriguing properties of MTD and Divergence Steering in reasoning and creative generation tasks.

2 Background

Prediction of Hidden States (PHi) The PHi method, introduced by Hermann et al. [2025], creates an information bottleneck [Tishby and Zaslavsky, 2015] within a sequence model in order to measure the complexity of its in-context computation. A PHi layer is inserted between a model's "bottom" and "top" layers. The model consists of the following modules: The **Bottom Layers** (B_β) are the initial Transformer blocks that process the input sequence embeddings. The **PHi Layer** contains three key components: (1) An **encoder** (q_ψ) that, at time step t, maps the hidden state g_t from the bottom layers to a posterior distribution over a latent variable z_t. This distribution is typically a diagonal Gaussian, similar to variational auto-encoders [Kingma and Welling, 2014, Rezende et al., 2014]. (2) A **decoder** (a_ξ) that reconstructs the hidden state, creating g'_t from a sample of the latent variable z_t. (3) An **autoregressive prior** that predicts the distribution of the current latent z_t using only the history of previous latents, $z_{<t}$. This can be implemented with a single Transformer block (M_μ) and two additional linear transforms: b_κ, which maps the inputs to M_μ to the right dimensionality, and p_χ, which maps the output of the transformer to a prior distribution. The **Top Layers** (T_τ) are the remaining Transformer blocks that process the sequence of reconstructed hidden states g'. Finally, we have the standard token **Embedding** (E_ϵ) and the **Output Layer** (O_ω) [1].

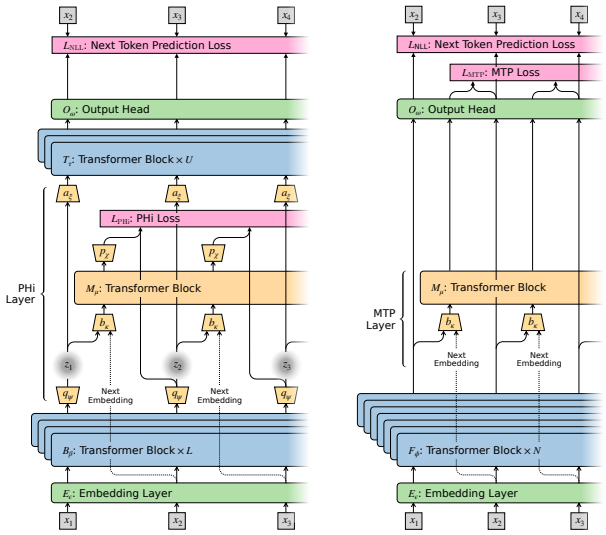


Figure 1: Comparison between the architecture of a PHi model (left) and of a MTP model (right).

The forward pass processing tokens $x_1, x_2, \ldots$ is described by these equations:

$$
\begin{aligned}
e_t &= E_\epsilon(x_t) && \text{Token embedding} \\
g_t &= B_\beta(e_1, \ldots, e_t) && \text{Output from bottom layers} \\
z_t &\sim q_\psi(\,\cdot\,|g_t) && \text{Latent sample from posterior} \\
g'_t &= a_\xi(z_t) && \text{Reconstruction of hidden state} \\
h_t &= T_\tau(g'_1, \ldots, g'_t) && \text{Output from top layers} \\
\pi(\,\cdot\,|x_{<t}) &= O_\omega(h_{t-1}) && \text{Next token prediction from output head} \\
L_{\text{NLL}}(t) &= -\log \pi(x_t|x_{<t}) && \text{Negative Log Likelihood (NLL) loss}
\end{aligned}
$$

The PHi loss (L_{PHi}) is the KL divergence between the posterior q_ψ, which has access to the current input x_t via g_t, and the prior p_χ, which only has access to past latents $z_{<t}$. We assume an initial latent z_0 is given.

$$
\begin{aligned}
c_t &= b_\kappa(z_{t-1}) && \text{Linear projection of last latent} && (1) \\
d_t &= M_\mu(c_1, \ldots, c_t) && \text{Output from PHi transformer block} \\
L_{\text{PHi}}(t) &= D_{\text{KL}}\Big(q_\psi(\,\cdot\,|g_t) \;||\; p_\chi(\,\cdot\,|d_t)\Big) && \text{PHi Loss} && (2) \\
&= D_{\text{KL}}\Big(q_\psi(\,\cdot\,|x_1, \ldots, x_t) \;||\; p_\chi(\,\cdot\,|z_0, z_1, \ldots, z_{t-1})\Big)
\end{aligned}
$$

[1] Here and in the remainder of the paper, Greek subscript letter indicate learnable neural network parameters.

The model is trained to jointly minimize both L_{NLL} and L_{PHi}. The PHi loss quantifies the information gain at each timestep—the amount of new useful information present in the current input token x_t that was not predictable from the history of latent states. It has been shown [Herrmann et al., 2025] that this value correlates well with the complexity and "interestingness" of tasks.

Multiple Token Prediction (MTP) Multiple Token Prediction (MTP) is a technique used to improve model performance and enable faster inference via methods like speculative decoding [Cai et al., 2024, Gloeckle et al., 2024, Liu et al., 2024, Xiaomi et al., 2025]. In this setup, a computationally cheap auxiliary module is trained to directly predict the main model's future output distribution.

First, consider a standard autoregressive model's forward pass:

$$
\begin{aligned}
e_t &= E_\epsilon(x_t) && \text{Token embedding} \\
h_t &= F_\phi(e_1, \ldots, e_t) && \text{Output of all main transformer blocks} \\
\pi(\,\cdot\,|x_{<t}) &= O_\omega(h_{t-1}) && \text{Next token prediction from output head} \\
L_{\text{NLL}}(t) &= -\log \pi(x_t|x_{<t}) && \text{NLL loss}
\end{aligned}
$$

The goal of MTP is to approximate the main model's prediction for the token one step further ahead, x_{t+1}. Note that often in MTP, there are additional modules that approximate predictions for tokens even further ahead, i.e., x_{t+n} for $n > 1$. We will not use them in this work. A separate, smaller MTP module (e.g., a single Transformer block M_μ) generates its own prediction without access to the full model's current hidden state h_t and is usually trained with a negative log-likelihood loss, which we call L_{MTP}. Optionally, the MTP module can be given access to the current token embedding e_t, as indicated by the square brackets.

$$
\begin{aligned}
c_t &= b_\kappa(h_{t-1}[, e_t]) && \text{Input to MTP module (projection of } h_{t-1} \text{and possibly } e_t) \\
d_t &= M_\mu(c_1, \ldots, c_t) && \text{Output from MTP Transformer block} \\
\pi_{\text{MTP}}(\,\cdot\,|x_{<t}) &= O_\omega(d_{t-1}) && \text{MTP's prediction for token } x_{t+1} \\
L_{\text{MTP}}(t) &= -\log \pi_{\text{MTP}}(x_t|x_{<t}) && \text{MTP Loss}
\end{aligned}
\tag{3}
$$

In order to predict the next token $x + 1$, the MTP module has access to the model's history via h_{t-1} (and optionally the current embedding e_t), but it crucially lacks the result of the main model's full computation at step t (i.e., h_t).

3 Multiple Token Divergence

Observe the similarity between the PHi framework's autoregressive prior p_χ and posterior q_ψ on the one hand, and the MTP prediction π_{MTP} and the full model prediction π on the other (Figure 1): in both cases, a computationally and informationally constrained module approximates the prediction from a full model. The key difference is that for PHi, this approximation occurs in a continuous latent space, whereas MTP operates directly on the discrete token distribution. Based on this analogy, we propose the *Multiple Token Divergence* (MTD) as an alternative to the PHi loss. It is defined as the KL divergence between the full model's next-token prediction π and the MTP module's prediction π_{MTP}:

$$
\begin{aligned}
L_{\text{MTD}}(t) &= D_{\text{KL}}\Big(\pi(\,\cdot\,|x_{\leq t}) \;||\; \pi_{\text{MTP}}(\,\cdot\,|x_{<t})\Big) \\
&= D_{\text{KL}}\Big(\pi\big(\,\cdot\,|F_\phi(e_1, \ldots, e_t)\big) \;||\; \pi_{\text{MTP}}\big(\,\cdot\,|F_\phi(e_1, \ldots, e_{t-1})[, e_t]\big)\Big).
\end{aligned}
\tag{4}
$$

The MTD loss, L_{MTD}, can either be optimized directly in conjunction with the standard next-token loss L_{NLL}, or it can be calculated post-hoc using an MTP module trained with L_{MTP} loss (see Section 4.2).

On the Difference between PHi and MTD While PHi introduced a powerful conceptual framework for analyzing a model's internal processing, its implementation can be complex. The stochasticity introduced by the variational information bottleneck can interfere with the main sequence prediction task. In contrast, MTD is significantly simpler to implement as it functions as a non-invasive auxiliary task; providing a less disruptive method to obtain similar insights into the model's per-token

computational effort. One interpretation of the PHi loss is that it measures, at every step, changes of the 'latent program' that the model synthesizes in-context to perform next-token prediction. The MTD loss, on the other hand, measures changes directly at the level of the output predictions. This distinction can lead to significant differences. For instance, a small change in the latent program could result in a large shift in the output predictions. As an illustrative example, consider a model trained on two distinct types of sequences: one type consists of uniformly random tokens, while the other is a specific single, repeated token. To distinguish between these two cases, the latent program only needs to gain one bit of information. The PHi loss would therefore be low. However, the resulting change in the output distribution is large—shifting from a uniform distribution to a one-hot distribution. In this scenario, the MTD can be as high as $D_{KL}(\text{one-hot}||\text{uniform}) = \log_2(\text{vocabulary size})$ bits. In such cases, we expect L_{MTD} to be significantly higher than L_{PHi}, an effect we observe in our experiments in Section 4.1. Conversely, one can imagine cases where the latent program changes significantly while the output predictions remain stable. However, the PHi training objective penalizes encoding such changes, as they do not sufficiently improve downstream predictions and thus represent an inefficient use of the information bottleneck. Whether the change in the latent program is larger than the one in the output distributions or not depends on the exact weighting of the PHi loss during training.

Access to the Latest Token Embedding An interesting nuance for both PHi loss and MTD is that the measured information gain at step t can originate from two distinct sources: (1) novel information contained within the current token x_t itself, and (2) complex computation performed by the model's main layers (B_β for PHi, F_ϕ for MTP), which cannot be easily approximated by the simpler prior or MTP module (M_μ). To disentangle these two sources, we can provide the prior/MTP module with direct access to the latest token embedding e_t, which is a common practice in MTP models (see Equation 3). This effectively isolates the second source of information gain. For the PHi framework, this modification involves updating Equation 1 to concatenate the previous latent state with the current embedding: $c_t = b_\kappa(z_{t-1}, e_t)$. With this change, the PHi prior has access to the same input token as the bottom layers, B_β. Consequently, the modified PHi loss, which we denote as $\hat{L}_{\text{PHi}}$, isolates the information gain attributable solely to the computation performed by B_β:

$$\hat{L}_{\text{PHi}}(t) = D_{\text{KL}}\Big(q_\psi(\,\cdot\,|x_1, \ldots, x_t) \;||\; p_\chi(\,\cdot\,|z_0, \ldots, z_{t-1}, e_t)\Big).$$

A PHi layer modified in this way acts as an information bottleneck that specifically measures computational effort. Information that the prior can easily extract from the input embedding e_t is allowed to pass freely, while information that is computationally non-trivial for B_β to extract is quantified by $\hat{L}_{\text{PHi}}$. The same logic applies to the MTD module when it is given access to the latest embedding. Arguably, PHi and MTD loss with access to the latest embedding provide a better measure of dense in-context computation. This access allows the prior/MTP module to account for trivial shifts in the predictions—like the ones described in the previous paragraph—thereby reducing the effective difference between the two metrics. In essence, providing access to the latest embedding allows us to quantify the information gain per step that is due to significant computational effort, whether measured in the latent space (PHi) or in the output distribution (MTD).

3.1 Decoding with Divergence Steering

So far, we have presented MTD as a post-hoc analysis tool. However, its formulation as the divergence between two output distributions provides a mechanism to influence the model's behavior during generation. It allows us to steer the decoding process towards or away from tokens that the shallow MTP module can easily predict. This gives rise to a novel decoding method. The core idea is to construct a new sampling distribution, s_α, by interpolating between the full model's prediction, π, and the MTP module's prediction, π_{MTP}. It is controlled by a single parameter, α: For $\alpha = 0$, we recover the original distribution from the full model: $s_0 = \pi$. For $\alpha = 1$, we use the distribution from the shallow MTP module: $s_1 = \pi_{\text{MTP}}$. For $\alpha < 0$, we extrapolate away from the MTP module's prediction. This amplifies the probability of tokens that are considered likely by the full model but unlikely by the shallow shortcut, effectively creating an 'anti-speculative' distribution biased towards computationally intensive tokens.

To perform this interpolation in a principled way, we travel along the geodesic path between the two distributions under the Fisher-Rao metric by mapping the distributions onto the positive orthant of a hypersphere and performing spherical linear interpolation [Miyamoto et al.,

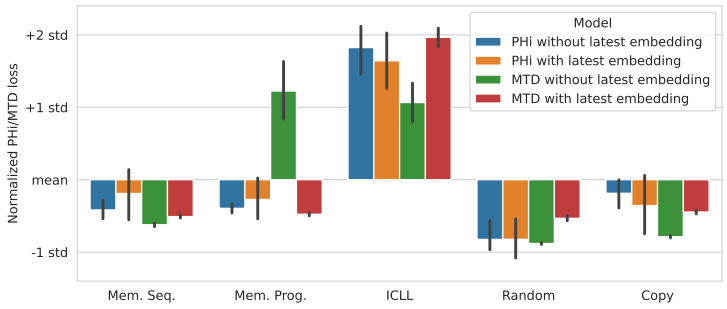


Figure 2: Normalized PHi or MTD loss of the four different model types on each of the five tasks. Only in-context language learning (ICLL) requires sophisticated in-context computation. This is reflected by the scores, with the exception of the MTD model without access to the latest embedding, which assigns high MTD also to the memorized programs task (see the discussion in Sections 3 and 4.1). Bootstrapped mean with 95% confidence intervals across 8 runs.

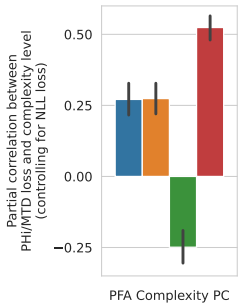


Figure 3: Partial correlation of PHi or MTD loss with the complexity of the modelled PFA, controlling for NLL. Also here, MTD without latest embedding access is the outlier.

2024]. Let $p = \pi$ and $m = \pi_{\text{MTP}}$ be two categorical distributions over a vocabulary of size K. Their representations on the hypersphere are the square roots of their probabilities, yielding $\mathbf{p}_g = (\sqrt{p_1}, \sqrt{p_2}, \ldots, \sqrt{p_K})$ and $\mathbf{m}_g = (\sqrt{m_1}, \sqrt{m_2}, \ldots, \sqrt{m_K})$. The angle between these two vectors is $\Theta = \arccos\left(\sum_k \sqrt{p_k m_k}\right)$. The geodesic path $\mathbf{s}_g(\alpha)$ between $\mathbf{p}_g$ and $\mathbf{m}_g$ is then given by $\mathbf{s}_g(\alpha) = \frac{\sin((1-\alpha)\Theta)}{\sin(\Theta)}\mathbf{p}_g + \frac{\sin(\alpha\Theta)}{\sin(\Theta)}\mathbf{m}_g$. To map this path back to a valid probability distribution $s(\alpha)$, we square each component of the vector $\mathbf{s}_g(\alpha)$, i.e., $s_k(\alpha) = (\mathbf{s}_{g,k}(\alpha))^2$.

This method introduces a new control knob, α, which is complementary to the standard temperature parameter, T. While T adjusts the entropy of the output distribution, α adjusts its "computational character." Because π_{MTP} often has higher entropy than π, changing α can also affect entropy. To isolate these effects, we can optionally project the interpolated distribution $s(\alpha)$ to a new distribution $\hat{s}_\alpha$ such that its entropy matches that of the original distribution, i.e., $H(s(\alpha)) = H(\pi)$ for all α. This provides two orthogonal levers for shaping the decoding process: T for entropy and α for computational density. Visualizations and additional details can be found in Appendix A. As we will show, the optimal choice of α is task-dependent (which is also the case for T): some tasks benefit from the robust, simpler predictions favored by positive α, while others may require the novel, less obvious paths uncovered by negative α.

4 Experiments

4.1 MTD and PHi Loss of Sequence Models Trained from Scratch

The considerations from Section 3 leave us with four different model configurations to compare: PHi and MTD models, each with and without access to the latest token embedding. The PHi model without this access corresponds to the original method proposed in prior work [Herrmann et al., 2025]. To compare these different setups, we train transformer models from scratch on several tasks and evaluate them in settings similar to those in Herrmann et al. [2025]. For details on the exact training setups, please see Appendix B.1.

Evaluation on Different Tasks The four model types are trained on five different tasks: (1) reciting memorized sequences, (2) modeling sequences from a small set of known formal languages (memorized programs), (3) in-context language learning (ICLL), where the formal language is unknown [Akyürek et al., 2024], (4) modeling random token sequences, and (5) a copying task that involves modeling random tokens where subsequences appear twice. Of these, only ICLL—which requires inferring the structure of an unknown probabilistic finite automaton (PFA) in-context—involves meaningful computation, in the sense that a non-trivial latent program must be synthesized by the model. Figure 2 shows a comparison of the normalized PHi and MTD losses for each task. The MTD with latest embedding access shows the clearest distinction between the one complex task and

the four "boring" ones; note the high value for ICLL and the consistently low values for all other tasks. For the MTD model *without* latest embedding access, we see the effect alluded to in Section 3: the loss is high for both ICLL and the memorized programs. For the memorized programs task, the actual in-context program required is minimal (only $\sim \log_2(10)$ bits to identify any one out of the ten memorized automata). However, the lack of information from the latest token causes a significant shift in the model's output distribution, resulting in a high MTD. Finally, giving the PHi layer access to the latest embedding does not appear to improve its ability to distinguish boring from interesting tasks.

Task Complexity Focusing on the ICLL task, we investigate the relationship between the models' PHi or MTD losses and the complexity of the underlying language, as measured by the description length of the PFA. Figure 3 displays the partial correlation between the mean PHi or MTD loss across a sequence and the language's complexity. We control for the mean NLL, as it is positively correlated with language complexity (r=0.367, 95% CI [0.315, 0.424]). Here again, we find that MTD with latest embedding access shows the strongest positive correlation (r=0.524 [0.480, 0.565]). In contrast, MTD without access to the latest embedding is negatively correlated with language complexity when controlling for NLL, confirming that it is not a reliable measure for this purpose. Further analysis (Appendix C) shows that only the original PHi loss (without embedding access) and the MTD loss with embedding access show a clear, positive token-wise relationship with language complexity after controlling for NLL.

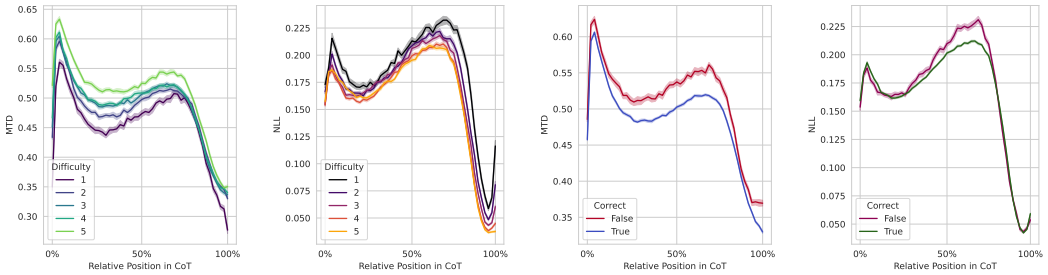
4.2 Pre-Trained Language Models

To validate our hypotheses on existing large-scale models, we leverage the pre-trained, open-source MiMo-7B model [Xiaomi et al., 2025]. We chose this model for two reasons. First, as a modern, high-quality 7B parameter model, its base pre-training incorporates an MTP objective, providing the built-in auxiliary prediction heads necessary for calculating the MTD without any post-hoc modification. Second, its compact size allows for the efficient, large-scale experimentation required to statistically validate our hypotheses across diverse tasks. We note that, while MiMo-7B was trained with an MTP objective from the outset, a similar setup could be achieved for other models by keeping the base model frozen and training an MTP head using standard teacher-student distillation [Schmidhuber, 1992, Hinton et al., 2015] with a fraction of the original data and compute.

Reasoning Difficulty We employ the MATH dataset [Hendrycks et al., 2021], which provides mathematics problems labeled from Level 1 (easy) to Level 5 (hard), along with detailed reasoning solutions. We first compute the mean MTD for the provided step-by-step solution for each problem in the dataset and find that it clearly correlates with the difficulty level (r=0.179, 95% CI [0.152, 0.203]). Interestingly, the NLL loss *negatively* correlates with problem difficulty (r=-0.249 [-0.274, -0.224]). This suggests that from the model's perspective, reasoning chains for difficult problems are no less plausible or predictable. However, the higher MTD indicates that the model makes increased use of its full capacity to process and generate them. We also have the model generate ten different chains-of-thought (CoTs) for each problem and repeat the analysis on these self-generated solutions. There again, we observe very similar results: the partial correlation between MTD and difficulty level, controlling for NLL, is r=0.199 [0.189, 0.208], while the correlation between NLL and difficulty is r=-0.158 [-0.168, -0.149]. These effects hold consistently across most problem categories, as shown in Figures 10 and 11 (Appendix). Since the provided rationales, as well as the generated CoTs, are longer for more difficult problems, the cumulative NLL also correlates positively with difficulty level (see Figures 12 and 13 in the Appendix).

We track the token-wise values for MTD and NLL across each generated CoT. As seen in Figure 4a, the positive correlation between MTD and problem difficulty holds consistently from the first tokens of the response to the last. Likewise, the negative correlation for NLL persists throughout the generation, even though not as pronounced (Figure 4b). The difference between MTD and NLL in their correlation with the problem difficulty is notable because, at a global level, MTD and NLL are positively correlated with each other (r=0.255 [0.246, 0.265]). This highlights that MTD captures a distinct signal related to computational effort that is not present in the standard NLL loss.

Reasoning Accuracy For the self-generated CoTs, we also investigate the relationship between MTD values and the correctness of the final answer. Figure 4c plots the token-wise MTD, stratified by whether the rationale was correct or incorrect. We observe that correct responses are consistently



(a) MTD vs. Difficulty (b) NLL vs. Difficulty (c) MTD vs. Correctness (d) NLL vs. Correctness

Figure 4: Token-wise losses against relative positions in self-generated CoT for the MATH test dataset. MTD shows a clear correlation with difficulty across the full CoT (a), the relationship between NLL and difficulty is less clear (b). Similarly, correct CoTs show higher MTD over all relative positions (c), which is not the case for NLL (d).

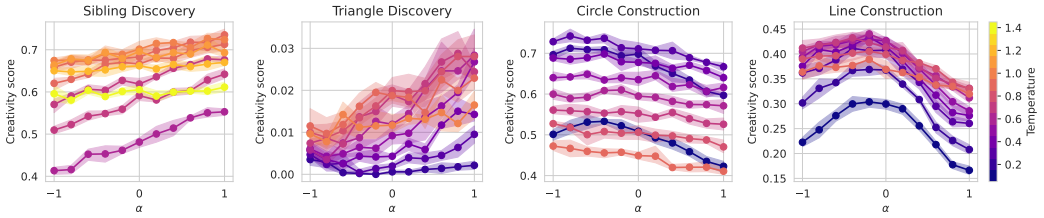


Figure 5: For the discovery tasks, positive α leads to higher creativity, whereas for the construction tasks, negative α leads to higher creativity. Results for geodesic distributions s_α.

associated with lower MTD. The relationship between NLL and correctness is less consistent (see Figure 4d). Following the methodology from prior work [Herrmann et al., 2025], we randomly assemble pairs of one correct and one incorrect CoT for each math problem. The probability of choosing the correct CoT when picking the one with the lower mean MTD is 67.1% (95% CI: [65.4%, 68.7%]). When selecting the one with the lower NLL, the probability is 73.3% [71.9%, 74.8%]. For the cases where NLL and MTD are agree, we get 80.4% [78.5%, 81.9%] accuracy. We repeat these experiments on the GSM-8k dataset [Cobbe et al., 2021], where we find that selecting CoTs with lower MTD yields 66.0% [62.9%, 69.2%] correct answers, while lower NLL yields 72.2% [69.1%, 75.0%] and combined yields 75.5% [71.7%, 79.2%]. For token-wise curves, please see Appendix C.1. These findings stand in contrast to the results for PHi loss, where, for a Llama 3B model, correct answers are associated with a *high* PHi loss [Herrmann et al., 2025]. We hypothesize that different models may have different tendencies to either overly simplify or overly complicate their reasoning process [Sui et al., 2025]. This could determine for a given model architecture or training regime whether computationally intensive answers are more or less likely to be correct.

4.3 Divergence Steering and Creative Tasks

Having established MTD as an indicator of complex in-context computation, we now investigate whether we can use it to influence model generation. Specifically, can biasing generation towards tokens with high MTD lead to more complex or creative outputs? The Divergence Steering method allows us to test this hypothesis. We adopt the creative algorithmic toy task framework proposed by Nagarajan et al. [2025], training transformer models on four distinct tasks: sibling discovery, triangle discovery, circle construction, and line construction (for details, please see Appendix B.3). The objective for each task is to generate sequences that are simultaneously *valid*, *novel* (i.e., not memorized from the training set), and *unique* within a fixed number of attempts. Success is measured by a creativity score, where 1 indicates perfect performance across all three criteria. All models used in this experiment are configured with MTP modules that have access to the latest token embedding. Figure 5 shows the creativity scores across a range of values for temperature and our steering parameter, α. The results reveal a task-dependent effect. For the "discovery" tasks, positive values of

α—which bias generation toward the simpler predictions of π_{MTP}—yield higher creativity scores. For the "construction" tasks, negative values of α—which create an "anti-speculative" distribution biased away from π_{MTP}—lead to better performance. A more detailed analysis (Appendix C.2) suggests that positive α helps the model avoid memorized solutions (improving novelty), whereas negative α can encourage the generation of more structurally sound outputs (improving validity). The optimal strategy, therefore, depends on the specific demands of the task. Crucially, temperature and α function as largely independent controls over the decoding process: for all four tasks, the best-performing combination of temperature and α achieves significantly higher creativity scores than optimizing for temperature alone. The qualitative behavior is similar for both geodesic and fixed-entropy distributions (see Figure 16 in the Appendix).

5 Discussion & Future Work

In our experiments, MTD outperforms PHi loss in differentiating "boring" from "interesting" tasks and simple from complex ones. It successfully isolates the per-token information gain attributable to non-trivial, or "irreducible" [Wolfram, 2002], computation by the model. However, the utility of the MTD signal is contingent on the relative capacities of the main model and the MTP module (M_μ): if the MTP module is too powerful, MTD approaches zero, and if it is too weak, MTD offers little beyond the standard NLL loss. Furthermore, because the shortcut module has fewer parameters, MTD may entangle genuine computational effort with memorization. Our findings also surface several intriguing questions. The positive correlation of MTD with problem complexity, in direct contrast to the negative correlation of NLL, warrants further investigation to determine if this is a general pattern across models and scales. Similarly, our result that lower MTD is associated with correct reasoning contrasts with prior findings for PHi loss, suggesting the relationship between computational effort and correctness is complex and model-dependent. While Divergence Steering enhanced performance on creative tasks, in preliminary experiments we found no clear improvement in the reasoning of large pre-trained models, perhaps because significant changes to the decoding strategy interfere with behaviors learned during post-training.

MTD and Divergence Steering have the potential for many applications in training and inference. Examples could be **Dynamic Compute Allocation:** MTD could be monitored in real-time during generation. A prolonged period of low MTD might trigger early stopping for a simple task, while a sudden spike in MTD could activate more powerful components (e.g., additional Mixture-of-Experts layers) for a difficult step. **Solution Convergence:** The transition from a high-MTD processing phase to a low-MTD conclusion could act as a signal that the model has "settled" on a solution, potentially allowing for more efficient decoding. **Intrinsic Motivation:** In agent-based settings, MTD could serve as an intrinsic reward. This would encourage an agent to pursue policies that lead to computationally interesting states (high information gain), fostering the development of more sophisticated behaviors. **Open Endedness:** MTD and Divergence steering allows the filtering or direct generation of "interesting" data. This may help to prevent model collapse when training on self-generated data and enable more creative, open-ended learning.

6 Conclusion

In this work, we introduce Multiple Token Divergence (MTD), a practical and direct measure for quantifying the computational effort of language models. By measuring information gain in the output distribution, MTD serves as a more robust and stable metric than prior methods that rely on latent state compression. We show that giving the auxiliary prediction module access to the latest token embedding allows MTD to specifically isolate the information gain attributable to non-trivial computation. Our findings demonstrate that MTD successfully distinguishes complex in-context reasoning from simpler tasks and reveals a nuanced relationship between computational effort and predictive loss. As a non-invasive and easily implemented metric, MTD provides a valuable new tool for analysis and evaluation. Furthermore, we introduce Divergence Steering, a novel decoding method that uses the MTD signal to actively steer the generation process towards either more or less computationally dense sequences. Shaping this "computational character" is complementary to the standard entropy adjustment using decoding temperature and improves creative generation.

References

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. In *Proc. Int. Conf. on Machine Learning (ICML)*, Vienna, Austria, July 2024.

C. H. Bennett. Logical depth and physical complexity. In *The Universal Turing Machine: A Half Century Survey*, pages 227–258. Oxford University Press, 1988.

Tom B Brown et al. Language models are few-shot learners. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Virtual only, December 2020.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv: 2401.10774*, 2024.

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*, abs/2110.14168, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. URL `https://doi.org/10.48550/arXiv.2407.21783`.

Eric Elmoznino, Tom Marty, Tejas Kasetty, Léo Gagnon, Sarthak Mittal, Mahan Fathi, Dhanya Sridhar, and Guillaume Lajoie. In-context learning and occam's razor. *ArXiv*, abs/2410.14086, 2024.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15706–15734. PMLR, 21–27 Jul 2024.

Peter D. Grünwald. *The Minimum Description Length Principle*. Springer, 2007.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Vincent Herrmann, Róbert Csordás, and Jürgen Schmidhuber. Measuring in-context computation complexity via hidden state prediction. In *Forty-second International Conference on Machine Learning*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Int. Conf. on Learning Representations (ICLR)*, Banff, AB, Canada, April 2014.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, USA, November 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Sachit Menon, David Blei, and Carl Vondrick. Forget-me-not! Contrastive critics for mitigating posterior collapse. In *Uncertainty in Artificial Intelligence*, pages 1360–1370. PMLR, 2022.

Henrique K Miyamoto, Fábio CC Meneghetti, Julianna Pinele, and Sueli IR Costa. On closed-form expressions for the fisher–rao distance. *Information Geometry*, 7(2):311–354, 2024.

Vaishnavh Nagarajan, Chen Henry Wu, Charles Ding, and Aditi Raghunathan. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. In *Forty-second International Conference on Machine Learning*, 2025.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 32, pages 1278–1286, Beijing, China, June 2014.

Jürgen Schmidhuber. Curious model-building control systems. In *Proc. Int. Joint Conf. on Neural Networks*, pages 1458–1463, 1991a.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. Int. Conf. on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227, 1991b.

Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992. doi: 10.1162/neco.1992.4.2.234.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, 2015.

Paul M Vitányi. Meaningful information. *IEEE Transactions on Information Theory*, 52(10), 2006.

Stephen Wolfram. A new kind of science. 2002.

LLM-Core Xiaomi, :, Bingquan Xia, Bowen Shen, Cici, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang, Huaqiu Liu, Jiebao Xiao, Jinhao Dong, Liang Zhao, Peidian Li, Peng Wang, Shihua Yu, Shimao Chen, Weikun Wang, Wenhan Ma, Xiangwei Deng, Yi Huang, Yifan Song, Zihan Jiang, Bowen Ye, Can Cai, Chenhong He, Dong Zhang, Duo Zhang, Guoan Wang, Hao Tian, Haochen Zhao, Heng Qu, Hongshen Xu, Jun Shi, Kainan Bao, Kai Fang, Kang Zhou, Kangyang Zhou, Lei Li, Menghang Zhu, Nuo Chen, Qiantong Wang, Shaohui Liu, Shicheng Li, Shuhao Gu, Shuhuai Ren, Shuo Liu, Sirui Deng, Weiji Zhuang, Weiwei Lv, Wenyu Yang, Xin Zhang, Xing Yong, Xing Zhang, Xingchen Song, Xinzhe Xu, Xu Wang, Yihan Yan, Yu Tu, Yuanyuan Tian, Yudong Wang, Yue Yu, Zhenru Lin, Zhichao Song, and Zihao Yue. Mimo: Unlocking the reasoning potential of language model – from pretraining to posttraining, 2025. URL `https://arxiv.org/abs/2505.07608`.

A Fixed Entropy Projection for Divergence Steering

To project the distribution s_α onto the hypersurface with entropy $H(p)$, we solve the following optimization problem:

$$\begin{aligned} \min_{\hat{s}_\alpha} \quad & D_{\mathrm{KL}}(\hat{s}_\alpha || s_\alpha) \\ \text{subject to} \quad & H(\hat{s}_\alpha) = H(p) \\ & \sum_i \hat{s}_{\alpha,i} = 1 \end{aligned}$$

By solving the Lagrangian, we see that this is equivalent to finding a temperature-scaled version of s_α. This means that $\hat{s}_\alpha$ takes the form:

$$\hat{s}_\alpha = \text{softmax}\left(\frac{\log s_\alpha}{T}\right)$$

for some temperature T, such that the entropy constraint $H(\hat{s}_\alpha) = H(p)$ is met. Since entropy is a smooth monotonic function of the temperature, we can use a fast root-finding algorithm like binary search to find the correct value for T. Figures 6 visualizes the interpolation process and the fixed entropy projection. Figure 7 shows the resulting distributions.

In practice, divergence steering, either with geodesic interpolation or this fixed-entropy projection, does not meaningfully slow down the generation process. For large vocabularies, however, it might be sensible use Divergence Steering in combination with top-k sampling and only optimize the remaining smaller distribution.

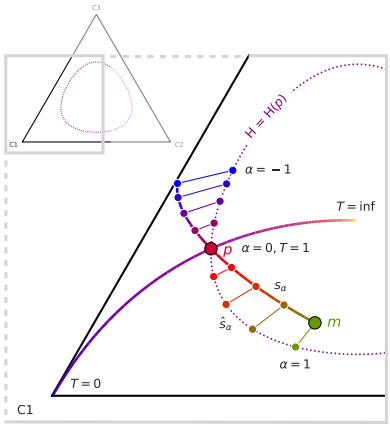


Figure 6: Divergence Steering on a K=3 simplex with temperature curve for p, geodesic interpolation between from m to p and beyond, and projection onto distributions with a fixed entropy of $H(p)$.

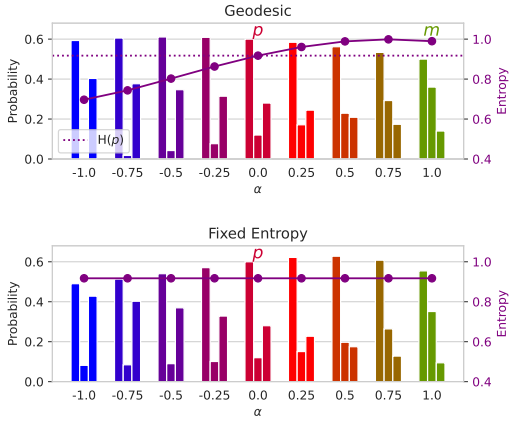


Figure 7: Distributions corresponding to Figure 6. Geodesic interpolation s_α, and the entropy of the resulting distribution (top). The same distributions projected onto the surface with fixed entropy, $\hat{s}_\alpha$ (bottom).

B Experiment Details

B.1 Sequence Models Trained from Scratch

We train all models using the Adam optimizer, a batch size of 16 and gradient norm clipping of 1.0. The learning rate is 0.0003, with a 500 step linear warm-up from zero and no decay. All losses are weighted equally, for the PHi loss we take the mean of the element-wise KL-Divergence for z, not the sum. Every model variation is trained 8 times with different random seeds for the initial weights and the procedurally generated data (which results in different memorized sequences and programs). The training of a model can be done on a single consumer-grade GPU (e.g., NVIDIA RTX 4090).

The base model is based on the Llama 3.2 architecture [Dubey et al., 2024].

- Number of layers: 12
- Model dimensionality: 768
- Number of attention heads: 6
- MLP intermediate size: 2048
- Embedding layer and output head are tied

PHi models:

To prevent posterior collapse, we employ an additional contrastive self-critic loss [Menon et al., 2022].

- Training steps: $30,000$
- Placement of the PHi Layer: After the 10th layer
- z dimensionality: 768
- q_ψ: Linear transform
- a_ξ: Linear transform
- b_κ: Linear transform
- M_μ: One transformer block like the ones in the rest of the model
- p_ψ: Linear transform

MTD models:

- Training steps: $10,000$
- b_κ: Linear transform
- M_μ: One transformer block like the ones in the rest of the model

For generation of training and testing data, we follow Herrmann et al. [2025]. The only difference is that we do not perturb any tokens during training, and that we use the same models for the task differentiation and task complexity experiments (Section 4.1).

B.2 Pre-Trained Language Models

For our experiments, we use the SFT version of the MiMo-7B model [Xiaomi et al., 2025]. To calculate the MTD, we use the included MTP head that predicts one token in advance.

All experimental results include bootstrapped 95% confidence intervals.

B.3 Divergence Steering and Creativity Tasks

The MTD models use the architecture and training procedure specified in in Section B.1. For each task, a dedicated model is trained for $50,000$ steps. No seed conditioning is used. For task definitions and evaluation procedure, we refer to Nagarajan et al. [2025].

The creativity score is defined as the fraction of all generated items that are valid, unique, and novel among. In addition, we define three more scores:

- Validity score: fraction of valid items among all generated items
- Uniqueness score: fraction of unique items among valid generated items
- Novelty score: fraction of novel items among valid unique generated items

These are be used in the additional empirical analysis in section C.2.

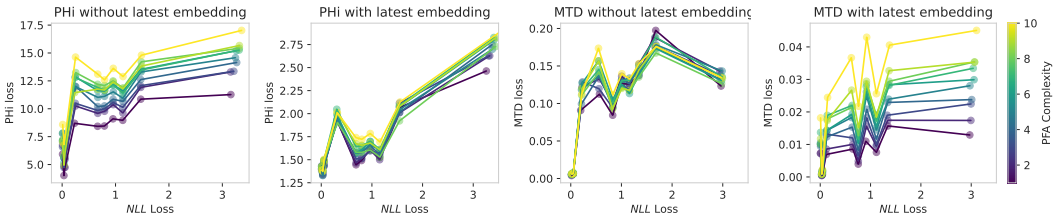


Figure 8: Token-wise PHi loss and MTD against binned NLL, for the different modeled PFA complexities. PHi loss without and MTD with latest embedding access both show a clear correlation with complexity level, across NLL bins.

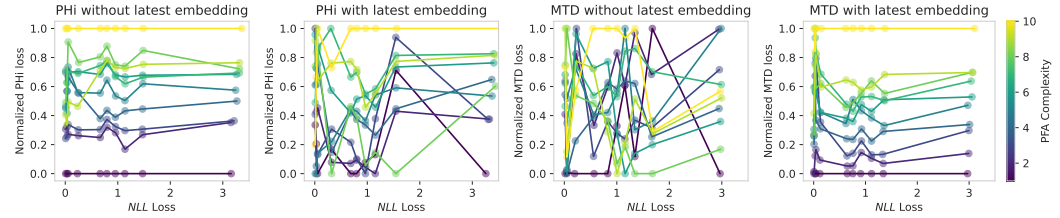


Figure 9: Similar to Figure 8, but normalized for each NLL bin. PHi loss without access to the latest embedding, and MTD loss with access to the latest embedding both show a clear correlation with complexity level, across NLL bins.

C Additional Experimental Results

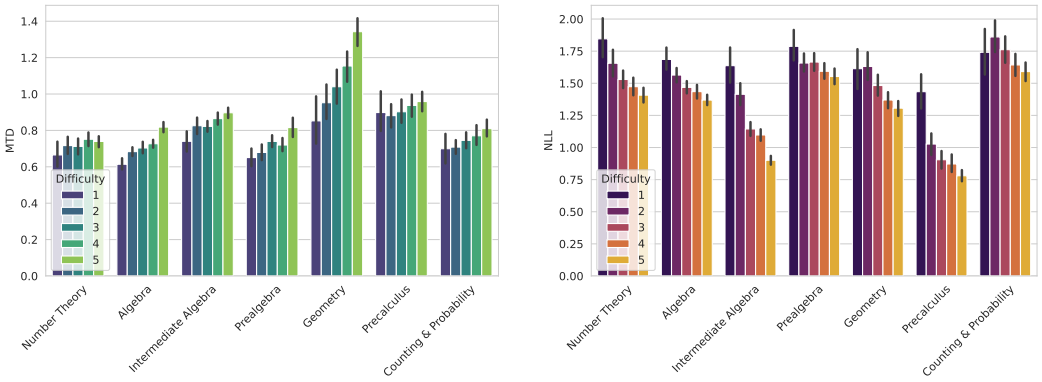
Figure 8 shows the token-wise PHi or MTD loss against binned NLL loss, broken down by PFA complexity (from 1, simple, to 10, complex). Figure 9 shows the same results normalized across NLL bins, making it clear to see that PHi without and MTD with access to the latest embedding show the clearest tokenw-wise relationship with PFA complexity.

C.1 Pre-Trained Language Models

Figure 10 shows MTD and NLL for the provided step-by-step solutions, broken down by category and difficulty level. Figure 11 shows the same for the self-generated CoTs. The results are qualitatively similar, even though, although the differences between categories for the CoTs are less pronounced.

Figures 12 and 13 use the cumulative instead of the mean losses. Due to the fact that the provided solutions as well as the generated ones grow in length as the problems become more difficult, cumulative NLL also correlates positively with difficulty level.

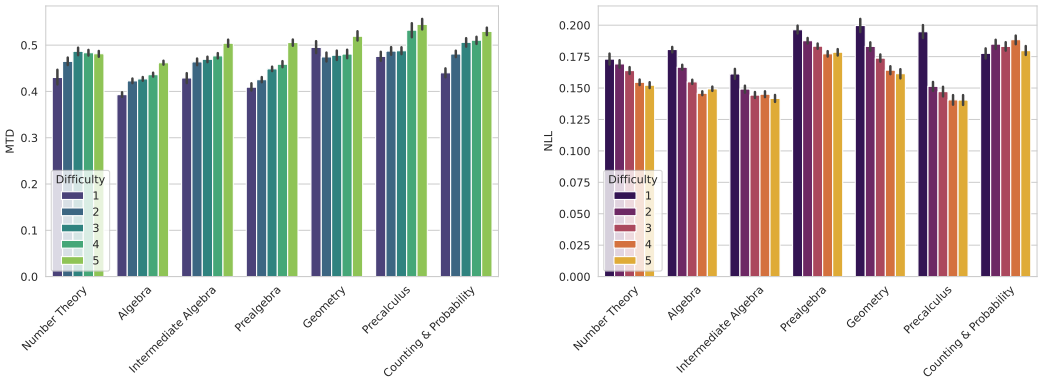
Figure 14 shows the development of MTD and NLL across self-generated CoTs for the problems of the GSM-8k test dataset (analogous to Figures 4c and 4d for MATH). Correct CoTs clearly have lower MTD, and lower NLL. Interestingly, for the GSM-8k dataset, the shapes of the NLL curves differ significantly from the shapes of the MTD, missing the prominent initial bump. Currently, we have no explanation for this.



(a) Mean MTD Loss

(b) Mean NLL Loss

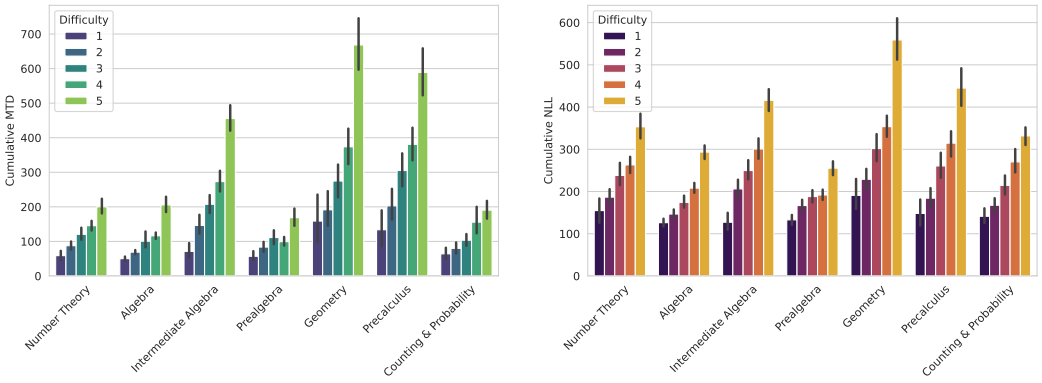
Figure 10: Mean losses of the MiMo model across the provided step-by-step solutions to the problems of the MATH test set, grouped by category and difficulty level. MTD clearly grows with difficulty, suggesting that the model is making more use of its computational capacity when processing more challenging problems. NLL loss, on the other hand, goes down with increasing complexity. Figure 11 shows similar results for self-generated chains of thought.



(a) Mean MTD Loss

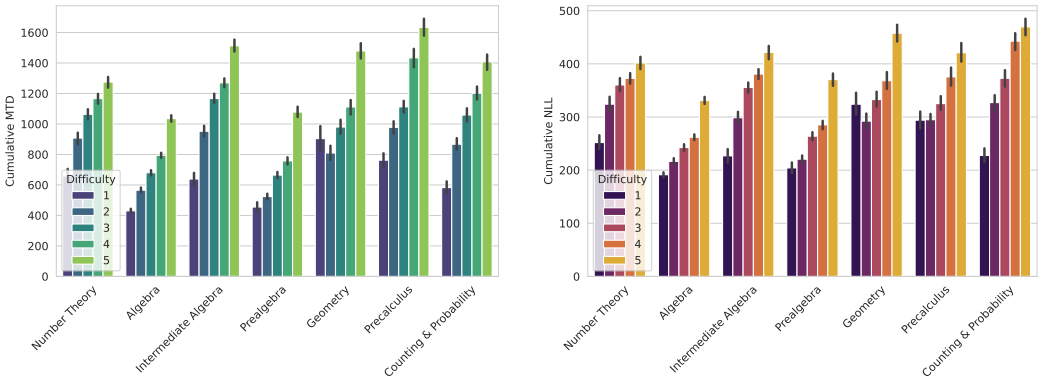
(b) Mean NLL Loss

Figure 11: Mean losses of the MiMo model across self-generated CoTs for the problems of the MATH test set, grouped by category and difficulty level. Similarly as in Figure 10, we observe that MTD clearly grows with difficulty, as the model is making more use of its computational capacity when generating the solutions to more challenging problems. Also here, the mean NLL goes down with problem difficulty.



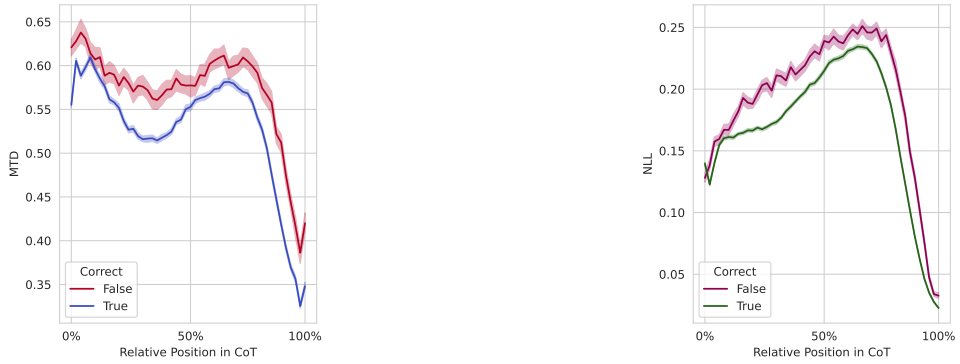
(a) Cumulative MTD Loss (b) Cumulative NLL Loss

Figure 12: Cumulative losses of the MiMo model across provided solutions from the MATH test set. Since more difficult problems have longer solutions, both cumulative MTD and cumulative NLL correlate with problem difficulty.



(a) Cumulative MTD Loss (b) Cumulative NLL Loss

Figure 13: Cumulative losses of the MiMo model across self-generated CoTs for the problems of the MATH test set. We observe a similar effect as in Figure 12.



(a) MTD vs. Correctness (b) NLL vs. Correctness

Figure 14: Token-wise losses against relative positions in self-generated CoTs for the GSM-8k test dataset. Lower MTD and lower NLL are both associated with more correct reasoning.

C.2 Divergence Steering and Creative Tasks

Figure 15 shows the creativity scores for the four tasks, using different values for temperature and α. In addition, we break down the results into validity, uniqueness and novelty scores. By the nature of the task, sibling and triangle discovery models are at risk of overfitting to the training data. A positive α value can help avoiding repeating memorized examples, as can be seen from the increased novelty scores. The models for circle and line construction, on the other hand, are less prone to overfitting, due to the combinatorial nature of the task. The novelty and uniqueness scores are consistently high. For these tasks, negative α appears to help construct increase the validity scores.

Figure 16 shows qualitatively very similar results for fixed entropy distributions.

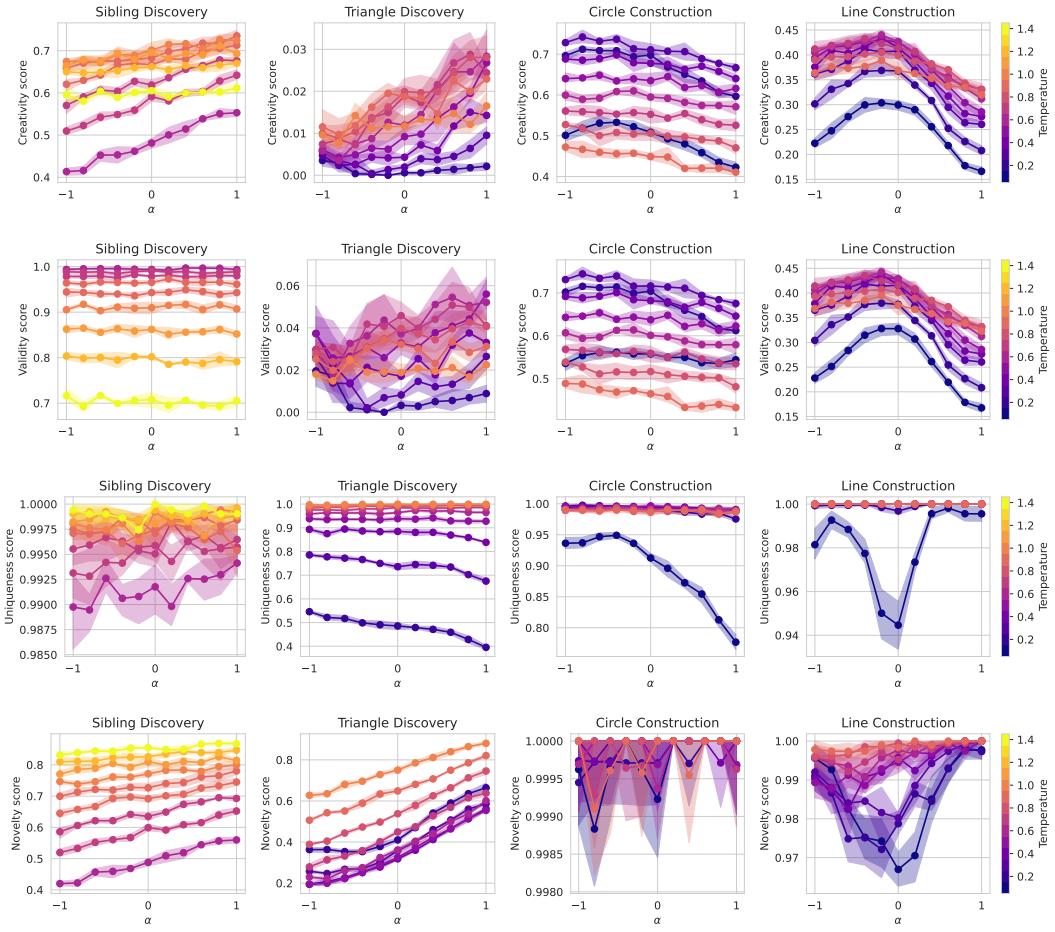


Figure 15: Breakdown of the creativity scores into validity, uniqueness, and novelty. Positive α can improve novelty, negative α can improve validity. Results for geodesic distributions s_α.

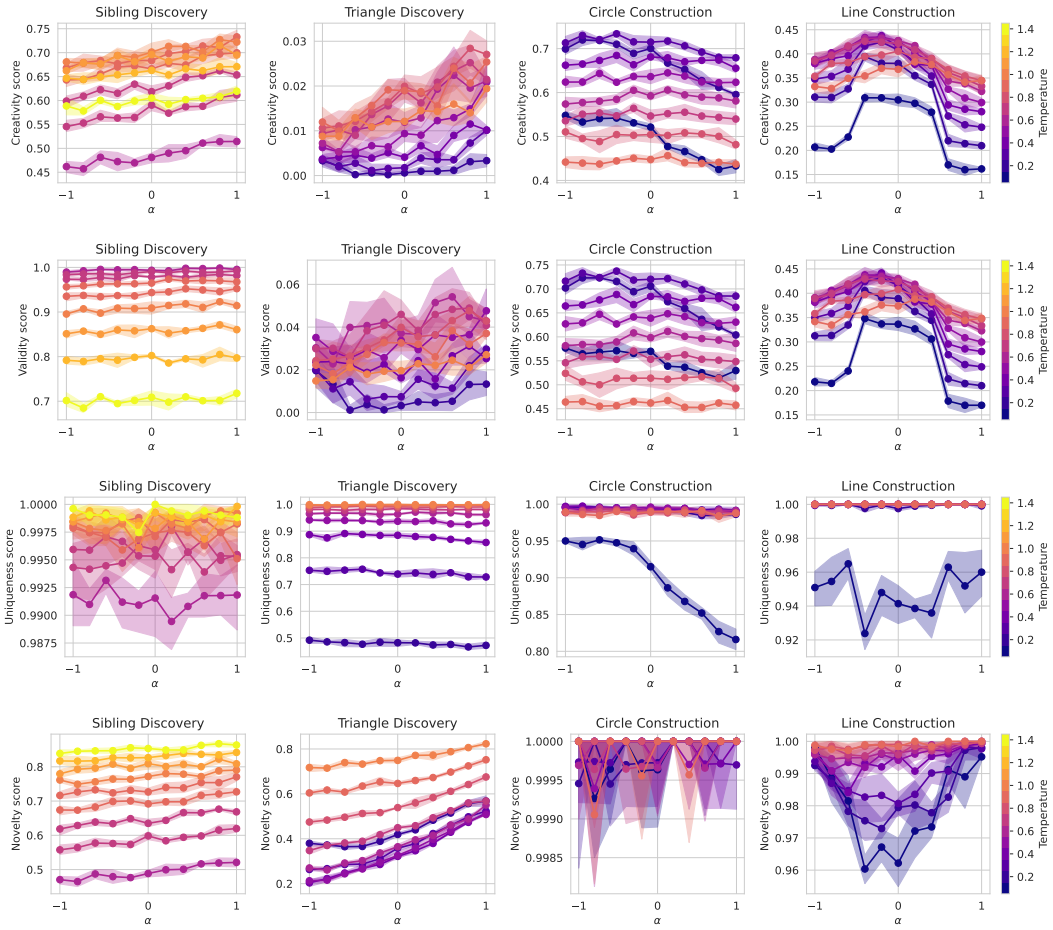


Figure 16: Breakdown of the creativity scores into validity, uniqueness, and novelty. Positive α can improve novelty, negative α can improve validity. Results for fixed entropy distributions $\hat{s}_\alpha$.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: All concrete claims are supported by empirical results.

   Guidelines:

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

2. **Limitations**

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

Answer: [Yes]

Justification: See Section 5

Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [NA]

Justification: No theoretical results in this paper.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The information in the experiments section and the appendix is sufficient to reproduce all results.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: See above, code will be made public upon acceptance.

   Guidelines:

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

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All experimental results include 95% confidence intervals

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The experiments are relatively small scale and can be run on a single GPU.

   Guidelines:

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

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification:

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA] .

    Justification: We expect no direct societal impact from this work.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA] .

    Justification:

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The model creates are given proper credit.

Guidelines:

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

13. **New assets**

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

Answer: [NA] .

Justification: No new assets in this paper.

Guidelines:

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

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

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

Answer: [NA]

Justification:

Guidelines:

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

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

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

Answer: [NA]

Justification:

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA] .

Justification:

Guidelines:

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