Next-Latent Prediction Transformers Learn Compact World Models

Jayden Teoh*, **Manan Tomar, Kwangjun Ahn, Edward S. Hu, Pratyusha Sharma, Riashat Islam, Alex Lamb and John Langford**

Microsoft Research

Abstract

Transformers replace recurrence with a memory that grows with sequence length and self-attention that enables ad-hoc look ups over past tokens. Consequently, they lack an inherent incentive to compress history into compact latent states with consistent transition rules. This often leads to learning solutions that generalize poorly. We introduce **Next-Latent Prediction** (NextLat), which extends standard next-token training with *self-supervised* predictions in the latent space. Specifically, NextLat trains a transformer to learn latent representations that are predictive of its next latent state given the next output token. Theoretically, we show that these latents provably converge to *belief states*, compressed information of the history necessary to predict the future. This simple auxiliary objective also injects a recurrent inductive bias into transformers, while leaving their architecture, parallel training, and inference unchanged. NextLat effectively encourages the transformer to form compact internal world models with its own belief states and transition dynamics—a crucial property absent in standard next-token prediction transformers. Empirically, across benchmarks targeting core sequence modeling competencies—world modeling, reasoning, planning, and language modeling—NextLat demonstrates significant gains over standard next-token training in downstream accuracy, representation compression, and lookahead planning. NextLat stands as a simple and efficient paradigm for shaping transformer representations toward stronger generalization.

1 Introduction

In learning theory, it is well known that simpler explanations of training observations tend to generalize better (Blumer et al. 1986; Langford and Schapire 2005). Modern transformers (Vaswani et al. 2017) stand in contrast to this principle. By replacing recurrence with a memory that scales with sequence length and self-attention that enables flexible look ups over past tokens, they achieve exceptional parallelization and predictive power. Yet, this very capability removes any inherent pressure to compress history into compact latent representations with consistent update rules. As a result, transformers often learn complex, task-specific shortcuts that fit the training data well but generalize poorly (Anil et al. 2022; Dziri et al. 2023; Liu et al. 2023; Wu et al. 2024). How can we encourage transformers to form simpler, more principled explanations that avoid such shortcuts? A natural approach is to reinstate a key property of recurrent models: the ability to learn *compact world models* that channel future prediction through compressed representations of the past. We will show that this inductive bias can be introduced while also retaining the parallel training efficiency of transformers.

In this paper, we introduce **Next-Latent Prediction** (NextLat), which extends the standard next-token prediction objective with self-supervised predictions in the latent space. NextLat jointly trains a transformer's latent representation alongside a latent dynamics model: the transformer encodes the history into compact latent states such that, when conditioned on the current latent and the next token (i.e., the "action"), the dynamics model can predict the transformer's next latent state. This objective drives the transformer to form compact latent summaries of past information with recurrent-like dynamics. In effect, NextLat enables the transformer to learn a compact internal world model while avoiding the sequential processing overhead of recurrent architectures. Importantly, NextLat leaves the transformer's architecture, inference, and parallel training procedure unchanged, introducing only a lightweight auxiliary loss on its latent representations during training. The approach is inspired by the *self-predictive learning* paradigm in reinforcement learning, a family of algorithms that learn representations by minimizing the prediction error of their own future latent states (Tang et al. 2023; Ni et al. 2024).

The core contributions of this paper are threefold. First, we establish a theoretical foundation showing that NextLat provably shapes transformer representations into *belief states*—compact summaries of past information sufficient for predicting future observations—a property enabling planning and generalization, yet absent in next-token prediction transformers. Second, we present a practical implementation of NextLat that preserves the transformer's architecture, inference procedure, and parallel training efficiency. By minimizing next-latent prediction error during training, we impose a structural preference that promotes compression and generalization, without adding parameters or computation at inference time. Finally, we empirically demonstrate NextLat's effectiveness across diverse domains spanning world modeling, reasoning, planning, and language modeling. Our results show that transformers trained with NextLat outperform stan-

*Correspondence to: t3ohjingxiang[at]gmail[dot]com

dard next-token training and other baselines in representation compactness, lookahead planning, and downstream accuracy. Together, these results position NextLat as an efficient and broadly applicable framework for learning compact, predictive, and generalizable representations in transformers.

2 Related Work

We are motivated by a long line of prior works in representation learning for prediction and control.

Self-Supervised Learning. Self-supervised learning (SSL; de Sa (1993); Balestriero et al. (2023)) is a framework for learning from unlabeled data, where a model generates its own supervisory signals from the structure of raw inputs. Across modalities such as vision, audio, and time series, SSL has proven highly effective for pretraining useful features, enabling downstream transfer that rivals, or even surpasses, models trained on labeled data (Liu et al. 2022; Balestriero et al. 2023; Zhang et al. 2024). There are several approaches to SSL. Our method falls under *self-predictive representation learning*, which jointly learns latent representations and a transition function that models how these representations evolve over a sequence. Self-prediction has driven state-of-the-art advances in reinforcement learning (RL) (Gelada et al. 2019; Zhang et al. 2020; Ye et al. 2021; Schwarzer et al. 2021; Hansen, Wang, and Su 2022). However, latent-space SSL remains underexplored in language modeling. A recent effort, LLM-JEPA (Huang, LeCun, and Balestriero 2025), minimizes distances between embeddings of paired text–code data, but relies on manually curated pairs and does not generalize to raw text. In contrast, our method introduces a fully self-supervised latent prediction objective that requires no paired data, making it **broadly applicable for training transformers across arbitrary sequence modeling domains**.

Belief States. In both sequence modeling and RL, models must reason over long histories of observations. To mitigate this curse of dimensionality, recent work focused on compressing history into latent representations capturing all information necessary for future prediction. In RL, this latent summary is formalized by Kaelbling, Littman, and Cassandra (1998) as a *belief state*, defined as: *"a sufficient statistic for the past history . . . no additional data about its past actions or observations would supply any further information about the current state of the world"*. In stochastic control, the same notion of sufficient statistics appears as "information state" (Striebel 1965). The idea of sufficient statistics is also key to learning state abstractions in RL (Li, Walsh, and Littman 2006). While recurrent neural networks naturally enforce such compression, transformers have no such constraint—their internal state, or memory, grows linearly with sequence length. Recently, Belief State Transformers (BST; (Hu et al. 2025)) extended the notion of belief states to transformers, and demonstrated benefits in planning tasks. Compared to BST, NextLat learns belief states without requiring a separate transformer and is much more computationally efficient. We compare these methods further in Section 5.

World Models. Loosely, a world model is an internal predictive model of how the world works, with varying interpretations across cognitive science (Craik 1967; Johnson-Laird 1983), neuroscience (Miall and Wolpert 1996; Friston 2010), and reinforcement learning (Sutton 1991; Schmidhuber 1990; Ha and Schmidhuber 2018). Whether transformer language models implicitly learn world models remains a debate; some studies report emergent world understanding (Patel and Pavlick 2022; Li et al. 2023; Gurnee and Tegmark 2024), while others find incoherent world structure (Vafa et al. 2024, 2025). However, successful approaches in learning world models such as *MuZero* (Schrittwieser et al. 2020) for achieving superhuman performance in video and board games, *Dreamer* (Hafner et al. 2019, 2021, 2025) for general model-based RL, and *Genie* (Bruce et al. 2024) for interactive video generation share a common principle: they explicitly learn a latent dynamics model that takes a latent state (an encoding of past observations), an action, and predicts the next latent state. Yet, learning these latent dynamics for autoregressive language modeling in transformers remains underexplored. Our method, NextLat, addresses this gap by introducing a self-prediction objective that explicitly learns a latent dynamics model which governs how the transformer's latent states evolve given new tokens (i.e., "actions"), thereby enabling the transformer to learn compact latent abstractions of the world with consistent dynamics.

Beyond Next-Token Prediction. In the domain of language modeling, a growing body of work has highlighted the myopic nature of the next-token prediction objective, which limits the model's capability in downstream tasks such as planning and reasoning (Bachmann and Nagarajan 2024; Nagarajan et al. 2025). Recent works have also found improvements from richer supervision signals that predict further into the future (Gloeckle et al. 2024; Liu et al. 2024; Hu et al. 2025; Ahn, Lamb, and Langford 2025). However, these approaches operate predominantly in the token space. NextLat takes a different approach: it shifts prediction into the latent space, enforcing self-consistent dynamics over the model's latent representations rather than its token outputs.

3 Methodology: Next-Latent Prediction

In this section, we introduce a simple, yet powerful, method for learning belief states in transformers through next-latent prediction (or more specifically, via *next-hidden state prediction*[1]). We begin by defining belief states in sequence modeling.

Definition 3.1 (Belief states in sequence modeling)**.** Let $X_{1:T}$ denote a token sequence $X_1, \ldots, X_T$. A random variable $\mathbf{b}_t = g(X_{1:t})$ is a *belief state* for the history $X_{1:t}$ if, for every bounded measurable function f of the future,

$$\mathbb{E}[f(X_{t+1:T}) \mid \mathbf{b}_t] = \mathbb{E}[f(X_{t+1:T}) \mid X_{1:t}] \quad \text{a.s.}$$

[1] In the sequence modeling literature, intermediate latent representations are often referred to as "hidden states". To disambiguate, we use the term "latent state" to broadly refer to learned representations within the transformer's residual stream, and "hidden state" to specifically refer to the final layer's output at each time step (i.e., the pre-logit activations).

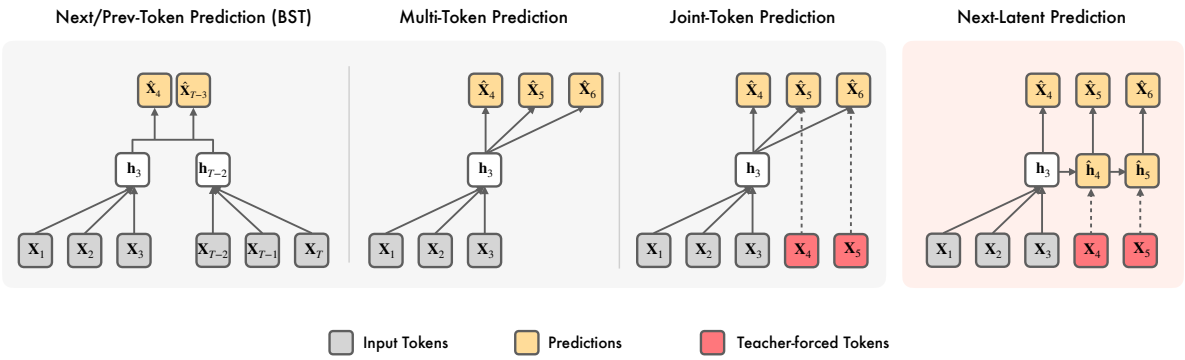


Figure 1: **Illustration of different predictive mechanisms** at time step $t = 3$. Other methods supervise only the token-level emissions, leaving intermediate latent representations implicit. In contrast, NextLat explicitly learns latent dynamics that predicts hidden state $\hat{h}_{t+1}$ from (h_t, x_{t+1}). Token-level supervision is then applied to the $\hat{h}_{t+1}$. Therefore, accurate multi-token predictions emerge as a consequence of faithful latent dynamics modeling, with the latent acting as the bottleneck.

Equivalently, $\mathbf{b}_t$ is a *sufficient statistic* (Striebel 1965) of the history $X_{1:t}$ for predicting the future tokens, i.e., from which we can sample from the distribution $\mathbb{P}(X_{t+1:T} \mid X_{1:t})$.

Why Next-Latent Prediction?

Theorem 3.2. *Consider the joint learning of three components:*

1. *a transformer with parameters θ that produces hidden states $\mathbf{h}_t$ at each time step t,*
2. *an output head p_θ modeling the next-token distribution, and*
3. *a latent dynamics model p_ψ modeling the transition dynamics of the transformer's hidden states.*

NextLat optimizes for the following consistency objectives:

Next-Token Consistency:

$$p_\theta(X_{t+1} \mid \mathbf{h}_t) = \mathbb{P}(X_{t+1} \mid X_{1:t}), \tag{1}$$

Transition Consistency:

$$p_\psi(\mathbf{h}_{t+1} \mid \mathbf{h}_t, X_{t+1}) = \mathbb{P}(\mathbf{h}_{t+1} \mid X_{1:t+1}), \tag{2}$$

where the right-hand side of Equation (2) is the transition law induced by the transformer's weights. For these consistency objectives to be satisfied, $\mathbf{h}_t$ must converge to a belief state for the sequence $X_{1:t}$.

Proof Sketch. A formal proof by backward induction is provided in Appendix B. Intuitively, optimizing for next-token (Equation (1)) and transition (Equation (2)) consistency ensures existence of measurable maps, i.e., p_θ and p_ψ, that allow recursive decoding of future tokens from $\mathbf{h}_t$:

$$\mathbf{h}_t \xrightarrow[\text{decode token}]{p_\theta} X_{t+1} \xrightarrow[\text{update state}]{p_\psi} \mathbf{h}_{t+1} \xrightarrow[\text{decode token}]{p_\theta} X_{t+2} \xrightarrow[\text{update state}]{p_\psi} \mathbf{h}_{t+2} \cdots \xrightarrow{p_\theta} X_T.$$

For these maps to exist, and be learned, $\mathbf{h}_t$ must jointly optimize toward a belief state—a sufficient statistic for the history to predict the future.

Remark. Optimizing only next-token consistency in standard transformers does not ensure that $\mathbf{h}_t$ converges to a belief state (c.f. Theorem 3 in Hu et al. (2025)). Intuitively, self-attention enables ad-hoc lookup of past tokens, so there is no pressure to compress all necessary information about the past into compact latent summaries at every time step.

Learning to Predict Next-Latent States

We now describe the practical implementation of NextLat, which augments standard next-token prediction with a self-supervised predictions in the latent space. Our NextLat implementation operates primarily on the hidden states (i.e., the final-layer outputs) as they provide compact, fixed-dimensional vectors through which gradients can be propagated through the entire transformer efficiently. As usual, we optimize the transformer and output head for next-token prediction (Equation (1)) using the cross-entropy loss:

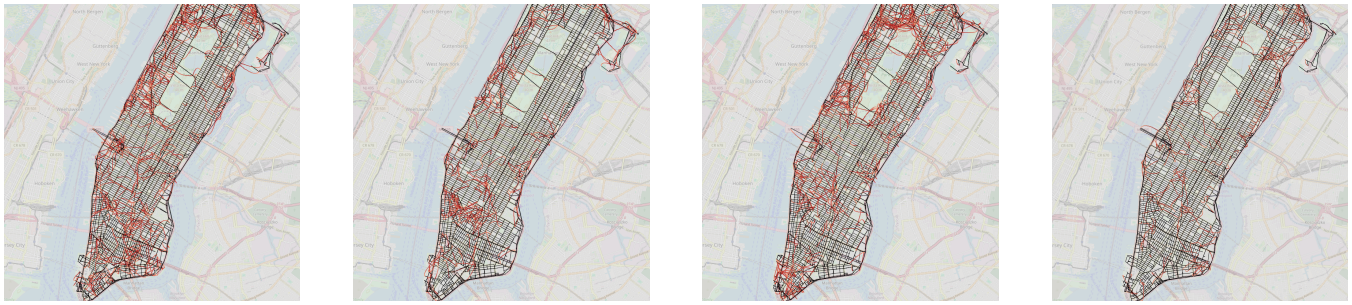
$$\mathcal{L}_{\text{next-token}}(\theta) = \mathbb{E}_{t<T}\big[-\log p_\theta(X_{t+1} \mid \mathbf{h}_t)\big].$$

NextLat additionally enforces transition consistency (Equation (2)) of the hidden states by introducing a latent dynamics model p_ψ that predicts the next hidden state $\mathbf{h}_{t+1}$ directly from $(\mathbf{h}_t, X_{t+1})$. For a deterministic transformer model, $\mathbb{P}(\mathbf{h}_{t+1} \mid \mathbf{h}_t, X_{t+1})$ is a Dirac distribution, and we can optimize p_ψ via regression. Moreover, observe that an ideal latent dynamics model should admit recursive consistency: its one-step map should compose correctly across multiple steps. Let $\hat{\mathbf{h}}_{t+d} = p_\psi(\mathbf{h}_t, X_{t+1:t+d})$ denote the recursive rollout of p_ψ over an d-step horizon using teacher-forced tokens $X_{t+1:t+d}$. We supervise all d intermediate rollouts using the smooth L1 loss[2]:

$$\mathcal{L}_{\text{next-h}}(\theta, \psi; d) = \mathbb{E}_t\Big[\frac{1}{d}\sum_{i=1}^{d} \text{SmoothL1Loss}\big(\mathbf{sg}[\mathbf{h}_{t+i}], \hat{\mathbf{h}}_{t+i}\big)\Big], \tag{3}$$

where $\mathbf{sg}[\cdot]$ denotes the stop-gradient operator, used to prevent representational collapse in self-predictive learning (Ni et al.

[2]Note that belief state convergence (i.e., Theorem 3.2) holds for any prediction horizon, including the 1-step case. Multi-step horizons are used only to provide richer supervision.



(a) GPT (b) MTP (c) JTP (d) NextLat (ours)

Figure 2: Reconstructed maps from transformers trained on Manhattan taxi rides using different objectives. Generated edges consistent with the true world model are colored black; invalid edges are red. Visibly, the transformer trained with NextLat learns a world model more consistent with reality.

2024). To further align the semantics of predicted states $\hat{\mathbf{h}}$ with true states, we introduce a complementary KL objective enforcing agreement in token prediction space:

$$\mathcal{L}_{\mathrm{KL}}(\theta, \psi; d) = \mathbb{E}_t\Big[\frac{1}{d}\sum_{i=1}^{d} D_{\mathrm{KL}}(p_\theta^{\mathbf{sg}}(\cdot \mid \mathbf{sg}[\mathbf{h}_{t+i}]) \parallel p_\theta^{\mathbf{sg}}(\cdot \mid \hat{\mathbf{h}}_{t+i})\Big], \quad (4)$$

where the output head $p_\theta^{\mathbf{sg}}(\cdot)$ is frozen so that gradients flow only through the latent dynamics model. This KL acts similarly to *knowledge distillation* (Hinton, Vinyals, and Dean 2015), providing soft supervision that guides learning of p_ψ. It also resembles *observation reconstruction* in self-predictive RL (Subramanian et al. 2022; Ni et al. 2024), encouraging $\hat{\mathbf{h}}_{t+i}$ to reproduce the distribution over next observations (i.e., the output head's logits).

Overall Objective. The final NextLat objective combines all components, minimizing the following loss:

$$\mathcal{L}_{\text{NextLat}}(\theta, \psi; d, \lambda_{\text{next-h}}, \lambda_{\mathrm{KL}}) = \mathcal{L}_{\text{next-token}}(\theta) + \lambda_{\text{next-h}}\,\mathcal{L}_{\text{next-h}}(\theta, \psi; d) + \lambda_{\mathrm{KL}}\,\mathcal{L}_{\mathrm{KL}}(\theta, \psi; d), \quad (5)$$

where $\lambda_{\text{next-h}}, \lambda_{\mathrm{KL}} > 0$ are scalar coefficients. Following standard training, we mask token-level losses (i.e., $\mathcal{L}_{\text{next-token}}$ and $\mathcal{L}_{\mathrm{KL}}$) corresponding to context or prompt tokens. However, we do not apply masking to the $\mathcal{L}_{\text{next-h}}$ objective, ensuring that belief state representations develop even during context processing. Importantly, during inference, the learned transformer decodes autoregressively as usual; p_ψ is used only during training to shape the transformer representations.

In the experiments that follow, we parameterize p_ψ using simple MLPs as our goal is to demonstrate that NextLat yields significant performance gains over baselines even without sophisticated latent dynamics architectures. More implementation details and a PyTorch-style pseudocode of the NextLat algorithm can be found in Appendix C.

4 Experiments

Modeling coherent latent dynamics and compact beliefs about the underlying data-generating process is fundamental to both algorithmic and human reasoning. Therefore, in this section, we evaluate NextLat on four key axes where such capabilities matter most: world modeling, reasoning, planning, and language modeling.

Our baseline comparisons include transformer-based belief-learning methods, i.e., BST (Hu et al. 2025) and JTP (Ahn, Lamb, and Langford 2025). Further discussions and detailed comparisons with these methods are provided in Section 5 and Appendix A. For completeness, we also report the performances of standard next-token prediction (GPT) and multi-token prediction (MTP). The MTP baseline follows the implementation of Gloeckle et al. (2024). Hereafter, we use the term "horizon" to refer to the multi-step prediction horizon d in JTP, MTP and NextLat, and we match horizon across these methods in all experiments to ensure fair comparisons. For specific experiment details such as hyperparameters, evaluation procedure, etc., please refer to Appendix D.

World Modeling: Manhattan Taxi Rides

Vafa et al. (2024) introduced a dataset of turn-by-turn taxi rides in Manhattan, where the true world model (i.e., the city's street map) is visually interpretable. Their study revealed that transformers trained on such trajectories can achieve near-perfect next-token accuracy, yet their internal maps remain incoherent; they reconstruct streets with impossible orientations and even flyovers above other roads.

Setup. We use the *random walks* dataset from Vafa et al. (2024), which consists of random Manhattan traversals (91M sequences, 4.7B tokens) between taxi pickup and dropoff points. Models are trained for 6 epochs (vs. 1 epoch in their study) as we observe performance does not converge within a single epoch. Due to its high computational cost, BST is excluded from this benchmark but included in our other experiments. For JTP, MTP, and NextLat, we set the multi-step prediction horizon at $d = 8$. We evaluate world-modeling performance using five comprehensive metrics:

1. Next-Token Test: Percentage of top-1 token predictions corresponding to legal turns under teacher-forcing on in-distribution validation sequences.
2. Valid Trajectories: Percentage of valid traversals for out-of-distribution (OOD) pickup–dropoff pairs.

	Next-Token Test (↑)	Valid Trajectories (↑)	Sequence Compression (↑)	Effective Latent Rank (↓)	Detour Robustness (↑)
GPT	100%	97.0%	0.65	160.1	85.0%
MTP	100%	98.1%	0.64	57.7	**95.0%**
JTP	100%	97.1%	0.32	215.8	87.0%
NextLat	100%	**98.7%**	**0.71**	**52.7**	**95.0%**
True world model	100%	100%	1.00	—	100%

Table 1: Comparison of transformers trained on Manhattan taxi rides with different objectives against the true world model across several metrics.

3. Sequence Compression: Percentage of cases where the model produces identical continuations when prompted with two different traversals arriving at the same state and sharing the same destination.
4. Effective Latent Rank: Effective rank/dimension of hidden states measured as the exponentiated Shannon entropy of the normalized singular values (Roy and Vetterli 2007); lower values indicate better compression.
5. Detour Robustness: Percentage of valid traversals for OOD pickup–dropoff pairs when we substitute the model's top-1 prediction with random detours (legal turns) 75% of the time.

The results are shown in Table 1. More details on training and evaluation are provided in Appendix D.

Results. Similar to the original study, all models achieved 100% accuracy on the next-token test. However, next-token accuracy is a limited diagnostic and cannot meaningfully assess the quality of a model's learned world model. In Figure 2, we visualize each model's internal map using the reconstruction algorithm proposed by Vafa et al. (2024). Visibly, the transformer trained with NextLat exhibits an internal map more consistent with the true world model. Although not perfect, its inconsistencies (red edges) are sparse and mostly local. Beyond this qualitative evidence, NextLat consistently outperforms all baselines across all metrics. On the trajectory validity and detour robustness metrics, NextLat demonstrates the **strongest generalization to OOD pickup–dropoff pairs**, even when random detours are introduced.

We analyze the compactness of the learned world models using two compression metrics. A model that accurately captures the underlying states and transitions should assign identical continuations to trajectories that end in the same state (i.e., intersection in Manhattan). By this criterion, NextLat achieves the **highest sequence compression of 0.71**. The true Manhattan graph comprises only 4,580 intersections and 9,846 edges, and therefore an effective world model should require only a modest latent dimensionality. Indeed, NextLat has the **lowest effective latent rank of 52.7—over 3x smaller than GPT's**. The combination of stronger planning performance and more compact latent representations reinforces the view that NextLat, by promoting belief state representations and coherent latent dynamics, enables transformers to learn substantially better world models—ones that are both accurate in their predictive structure and efficient in their internal representation of the environment.

Reasoning: Countdown

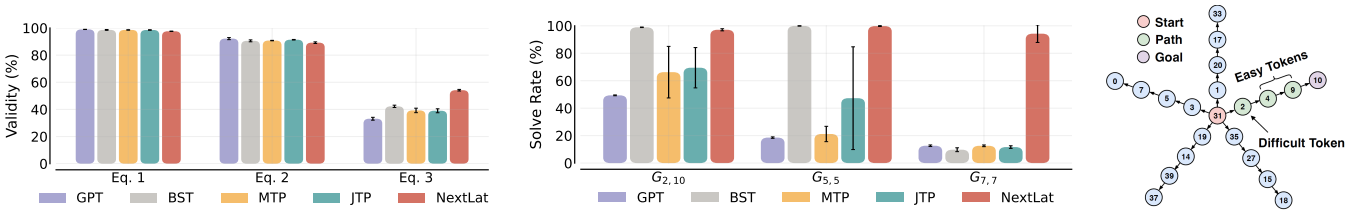
Model	Horizon (d)	Accuracy (%)
GPT	–	33.1
BST	–	42.3
MTP	1	39.2
	4	49.7
	8	57.3
JTP	1	39.0
	4	49.4
	8	55.0
NextLat	1	54.2
	4	<u>57.4</u>
	8	**57.8**

Table 2: Performance on Countdown. Best result is **bolded**, and second best is <u>underlined</u>.

Countdown (Countdown 2025) is a mathematical reasoning task and a generalized version of the Game of 24, which even frontier models such as GPT-4 (Achiam et al. 2023) have struggled with, achieving 4% by default (Yao et al. 2023). The goal of the task is to combine a set of given numbers with basic arithmetic operations $(+, -, \times, \div)$ to obtain a target number. For example, given the numbers $\{90, 8, 20, 50\}$, the target number 24 can be obtained using the following sequence of equations: $90 \times 8 = 720,\ 50 - 20 = 30,\ 720 \div 30 = 24$. Countdown poses a difficult combinatorial search problem due to its large branching factor and the need to efficiently explore the solution space to reach the target number.

Setup. Following Gandhi et al. (2024), we generate 500k training problems with target numbers ranging from 10 to 100 and reserve 10% of the targets for out-of-distribution evaluation. During both training and testing, we insert eight 'pause tokens' (Goyal et al. 2023) after the target number, allowing models additional computation to plan before generating a solution. Performance is measured as the percentage of 10k test problems for which a model produces a valid sequence of equations that correctly reaches the target number, averaged over three random seeds per baseline.

Results. As shown in Table 2, NextLat consistently outperforms all baselines on the Countdown task. Notably, even at



(a) (b) (c)

Figure 3: (a) Validity of equations (i.e., LHS = RHS) generated on Countdown. All models in this plot use $d = 1$. (b) Accuracy on Path-Star Graph task. Unlike the baselines, NextLat maintains close to 100% solve rate for all graphs. (c) Illustration of a $G_{5,5}$ Path-Star graph (Bachmann and Nagarajan 2024).

a shallow prediction horizon of $d = 1$, NextLat substantially surpasses the MTP and JTP baselines with the same horizon (**38% improvement**). To better understand this gap, we analyzed the equations generated by each model and evaluated their equation validity, i.e., whether the computed left-hand side equals the right-hand side. As shown in Figure 3a, most calculation errors occur in the final equation (Eq. 3). This indicates that the model's lack of planning capability results in its realization of being unable to achieve the goal only at the end. Unable to revise earlier missteps, it forces an invalid final equation to match the desired outcome—a behavior termed *the regretful compromise* by Ye et al. (2025). NextLat demonstrates **stronger lookahead planning**: even with $d = 1$, it achieves substantially higher mean validity in the final equation (54.2%) compared to the next best baseline (42.3%). This suggests that the latent-state prediction objective enables it to anticipate long-horizon dependencies and form globally consistent plans, reducing the tendency to make myopic errors. NextLat achieves the leading performance with horizons $d = 4$ and 8.

Planning: Path-Star Graph

A path-star graph (Bachmann and Nagarajan 2024) $G_{d,\ell}$ consists of a center node and d disjoint arms, each consisting of $\ell - 1$ nodes. Figure 3c depicts an instantiation of the $G_{5,5}$ topology. A training instance is a sequence that contains the edge list, the start and end nodes, and the correct path from start to end. This task represents a minimal instance of lookahead planning, a core capability underlying more complex behaviors such as storytelling. Yet, despite its apparent simplicity, next-token prediction models struggle to solve it.

Setup. Following Bachmann and Nagarajan (2024), we generate 200k training samples and set $N = 100$, such that node values in each graph are randomly drawn from $1, \ldots, 100$. For MTP, JTP, and NextLat, we set the multi-step prediction horizon to $d = \ell - 2$, ensuring that the target (end) node lies within the multi-step prediction horizon of the center node. We evaluate performance across three graph configurations: $G_{2,10}$, $G_{5,5}$, and $G_{7,7}$ across five random seeds per baseline.

Results. As shown in Figure 3b, NextLat maintains close to 100% solve rate for all topologies of the path-star graphs. BST, while able to solve $G_{2,10}$ and $G_{5,5}$, begins to fail at the larger graph $G_{7,7}$. The Path-Star task is designed to reveal the myopic behavior of teacher-forced next-token prediction models, which tend to exploit local shortcuts instead of learning to perform lookahead planning necessary to solve the task. This phenomenon, termed the *Clever Hans cheat* (Bachmann and Nagarajan 2024), has motivated methods such as BST, MTP, and JTP, that attempt to mitigate shortcut learning through multi-token predictions. However, these approaches still operate in token space, making them susceptible to local n-gram regularities that do not capture the underlying transition structure required for long-horizon planning. By contrast, NextLat performs prediction in latent space, enforcing transition consistency at the representation level. NextLat's success across all graph configurations, unlike MTP, JTP, and BST, suggests that latent-space prediction better avoids shortcut learning and yields more generalizable solutions.

Language Modeling: TinyStories

Next, we compare the language modeling capabilities of the models on TinyStories (Eldan and Li 2023), a dataset consisting of synthetic short stories. TinyStories aims to represent key challenges in text generation while keeping training tractable for small to medium scale models. After pretraining the transformer methods on the dataset, we train linear probes on the hidden representations of the frozen transformers to predict future tokens. This allows us to assess whether the models' representations capture belief-state–like abstractions that encode information predictive of future tokens, or just local token correlations.

Setup. Following Hu et al. (2025), we tokenize the dataset of 2.7 million stories into a vocabulary of 1,000 tokens and construct training sequences of length 256. All models are trained for 100k steps, which is sufficient for convergence. We include comparisons of transformers trained using MTP, JTP, and NextLat with multi-step prediction horizons $d \in \{1, 8\}$. After training, we freeze the model parameters and train 20 independent linear probes, one per token offset, to predict tokens at offsets $1, \ldots, 20$ steps ahead from the hidden states of the frozen models using the same dataset.

Results. We plot the difference in probe performance relative to probes trained on GPT's hidden states in Figure 4. For clarity, we display only selected token offsets here (see Figure 6 in the appendix for full results). Observe that the

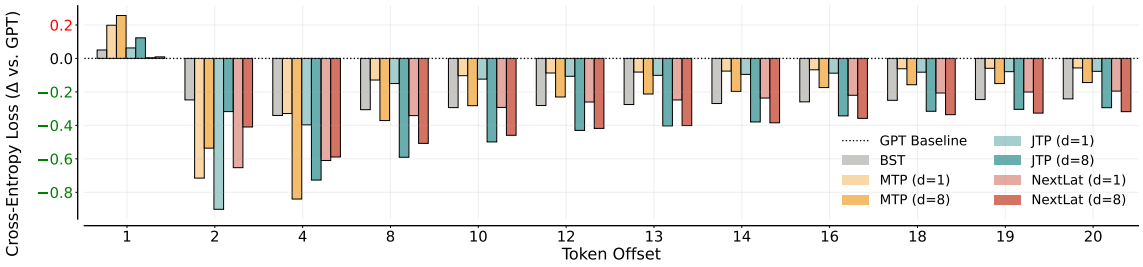


Figure 4: Cross-entropy loss difference relative to GPT, obtained from linear probes trained on frozen hidden states to predict tokens at varying token offsets (x-axis values) ahead. Lower values indicate better predictive performance.

additional token-level prediction objectives in BST, MTP, and JTP consistently cause significant degradation in next-token prediction (i.e., token offset = 1). Preserving high fidelity in next-token prediction is arguably important, as empirical studies show that lower perplexity correlates strongly with improved downstream performance (Gadre et al. 2024; Thrush, Potts, and Hashimoto 2025). Moreover, probe performance on JTP and MTP representations declines sharply with increasing token offset. This indicates that these multi-token prediction models, lacking guarantees of learning belief states, could encode information useful only for short-horizon prediction. In contrast, NextLat matches GPT's next-token performance across both $d \in \{1, 8\}$ and exhibits the strongest long-horizon predictive capability (up to 20 tokens ahead) for both $d = 1$ and $d = 8$. These results suggest that NextLat's latent-state objective induces belief-like representations that encode predictive information about future events—an ability essential for maintaining coherence in long-range narrative generation tasks like TinyStories.

5 Discussions

Comparison Against Current Approaches

In this section, we discuss the advantages of NextLat compared to prior transformer learning approaches, focusing discussions on belief-learning methods, i.e., BST and JTP. For readers unfamiliar with these methods, we include brief descriptions of their training objectives in Appendix A. Table 3 summarizes the training and inference parameters, training speed (in iterations per second), and gradient signal characteristics for each method on TinyStories, providing context for the discussions that follow.

Computational Costs. While BST benefits from $O(T^2)$ gradient signals per token sequence, this is a double-edged sword. Even with the optimized implementation of Hu et al. (2025), training remains extremely costly because gradients must be accumulated over all $O(T^2)$ predictions of different prefix–suffix pairs. Moreover, BST trains two transformer encoders, further increasing compute and parameter cost. During inference, BST also uses both transformer encoders, one for generation and the other for scoring the likelihood of the generated sequence. On TinyStories, BST is over $10\times$ slower than NextLat (see Table 3) in training speed. In contrast, NextLat with $d = 1$ incurs only a minor slowdown relative to GPT (3.26 vs. 3.72 iterations/s) while achieving the same belief-state learning guarantees as BST.

We next compare the compute costs of the multi-step prediction methods, i.e., MTP, JTP, and NextLat. The MTP implementation of Gloeckle et al. (2024), as well as other variants such as Liu et al. (2024), require additional transformer layers as the prediction horizon d increases, whereas JTP and NextLat keep parameter counts fixed across horizons. All three methods exhibit similar training speeds for $d = 1$. However, at $d = 8$, JTP is faster than NextLat because NextLat sequentially unrolls its latent dynamics model p_ψ to compute multi-step losses, while JTP computes them in parallel. Nonetheless, this modest increase in computation for NextLat yields substantially better performance than JTP across all benchmarks. Importantly, NextLat's sequential computation remains far more efficient than that of recurrent neural networks, which we discuss in Section 5.

Belief State Learning. GPT and MTP lack any theoretical learning pressure to form belief-state representations, which means that they do not necessarily learn sufficient representations predictive of future observations. JTP, in contrast, can learn belief states but only under the restrictive condition that the prediction horizon satisfies $d \geq k$, where k denotes the observability horizon of the underlying data-generating process (see Theorem A.1 in appendix). In long-context sequence modeling, however, the underlying process is often k-observable for a very large k, rendering this condition impractical. NextLat, on the other hand, learns belief-state representations independently of d and larger prediction horizons are used only to provide richer gradient signals. It also avoids the expensive $O(T^2)$ gradient computation required by BST to learn belief states. Taken together, these properties position NextLat as a simple, efficient, and theoretically grounded alternative to existing belief-state learning approaches (i.e., BST and JTP) in autoregressive sequence modeling.

Myopic Nature of Token-level Predictions. Token-level supervision is often myopic. Next-token prediction transformers tend to prioritize short-term dependencies, and studies have shown that early training can often resemble n-gram modeling, which can delay or even prevent learning of the true Markov kernel (Edelman et al. 2024; Makkuva et al. 2025). Bachmann and Nagarajan (2024) further showed that the myopic nature of next-token prediction training can trap models in suboptimal local minima, undermining long-

	GPT	BST	MTP	JTP	NextLat
Training parameters ($d = 1$ / $d = 8$)	57M	114M	64M / 114M	60M	66M
Inference parameters	57M	57M / 114M	57M	60M	57M
Training iterations/second ($d = 1$ / $d = 8$)	3.72	0.19	3.12 / 1.81	3.33 / 2.61	3.26 / 1.89
Gradients	$O(T)$	$O(T^2)$	$O(Td)$	$O(Td)$	$O(Td)$

Table 3: Comparison of training speed, parameter count and gradient signals provided on TinyStories for all methods. For training parameter numbers and training iterations/second, we report values for $d = 1$ and $d = 8$. If only a single value is shown, it means that the values are the same for both horizons. All training speeds are measured on a single NVIDIA H100 NVL GPU.

horizon planning. In our experiments, we find that adding additional token-level prediction objectives, as in BST, JTP, and MTP, not only degrades next-token performance but also fails to yield consistent gains in generalization. In contrast, NextLat, which emphasizes latent transition modeling as its primary objective, avoids degradation in next-token performance and improves downstream generalization by encouraging the learning of structured, predictive representations rather than shallow token-level correlations. In doing so, NextLat offers a compelling alternative to conventional token-level prediction objectives for transformer training.

NextLat vs. Recurrent Neural Networks

Algorithmic reasoning requires capabilities most naturally understood through recurrent models of computation, like the Turing machine. However, strict recurrence imposes a sequential computation bottleneck at training time. NextLat introduces a recurrent *inductive bias* via latent transition prediction without turning the transformer into a strictly sequential model.

Computational Efficiency and Relation to Backpropagation Through Time. RNNs incur $O(T)$ sequential dependence for training. In contrast, NextLat introduces only an additional sequential cost proportional to the rollout horizon $d \ll T$ during training, corresponding to the unrolling of the latent dynamics model p_ψ over d steps. This enables NextLat to largely retain the transformer's parallel training efficiency while also incorporating a recurrent inductive bias. In practice, this yields a controllable trade-off: the horizon d can be tuned to balance supervision strength and computational cost, without incurring the full sequential burden of RNNs.

Conceptually, the one-step and multi-step prediction in NextLat resembles truncated backpropagation through time (TBPTT) in RNNs, with truncation windows of 1 and d, respectively. However, a key distinction lies in how gradients are propagated. In TBPTT for RNNs, gradient computation is truncated beyond the chosen window, yielding biased gradient estimates that lack the convergence guarantees of stochastic gradient descent. In contrast, the theoretical convergence of belief-state learning in NextLat, as shown in Theorem 3.2, is independent of d, since convergence requires only one-step prediction optimality. Intuitively, NextLat performs full backpropagation through the transformer's computation graph, whose self-attention connects all past tokens, while the latent dynamics model is unrolled externally in an "outer loop". This outer-loop supervision does not truncate gradient flow within the transformer, and therefore avoids bias with respect to the chosen d. Larger prediction horizons simply provide richer supervision and faster empirical convergence.

Expressivity of the Recurrence. Modern state-space models (SSMs), such as S4 and Mamba, implement efficient linear recurrent updates in their hidden states, offering long-range dependency modeling while remaining highly parallelizable (Gu, Goel, and Ré 2022; Gu and Dao 2024). In contrast to SSMs, the latent dynamics model p_ψ in NextLat can express either linear or nonlinear transitions, depending on the choice of model architecture. Moreover, NextLat does not explicitly perform recurrence in the forward computation. Instead, recurrent-like dynamics emerge implicitly through one-step or multi-step unrolling of the latent dynamics model and aligning successive hidden states via regression. This setup allows NextLat to induce structured temporal consistency within the latent space without changing the transformer's architecture. However, it is important to note that NextLat modifies the learned representations, not the underlying circuit complexity. The overall computational expressivity of a transformer trained with NextLat remains bounded by that of the transformer itself, that is, by the TC^0 class of constant-depth threshold circuits, which upper-bounds the formal languages recognizable by fixed-depth transformers (Merrill, Sabharwal, and Smith 2022).

6 Conclusion

In this paper, we introduced Next-Latent Prediction (NextLat), a simple yet powerful framework that augments next-token training with self-supervised latent-space prediction, enabling transformers to learn belief-state representations. Theoretically, we show that the pressure to form concise latent summaries of past information sufficient to predict the future arises naturally from the NextLat objective. Empirically, NextLat yields more compact, predictive, and generalizable representations across tasks in world modeling, reasoning, planning, and language modeling—all without changing the transformer's architecture, parallel training efficiency, or inference procedure. By reintroducing a recurrent inductive bias through self-predictive latent dynamics, NextLat unifies the inherent bias toward compact and temporally-consistent representations of recurrent models with the scalability and parallelism of transformers. Our method is broadly applicable for training transformers in autoregressive sequence modeling domains. Looking ahead, we view NextLat as a step toward training objectives that endow autoregressive sequence models with simpler, more compact, and therefore more generalizable representations of complex systems.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ahn, K.; Lamb, A.; and Langford, J. 2025. Efficient Joint Prediction of Multiple Future Tokens. arXiv:2503.21801.

Anil, C.; Wu, Y.; Andreassen, A.; Lewkowycz, A.; Misra, V.; Ramasesh, V.; Slone, A.; Gur-Ari, G.; Dyer, E.; and Neyshabur, B. 2022. Exploring Length Generalization in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 38546–38556. Curran Associates, Inc.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. arXiv:1607.06450.

Bachmann, G.; and Nagarajan, V. 2024. The Pitfalls of Next-Token Prediction. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2296–2318. PMLR.

Balestriero, R.; Ibrahim, M.; Sobal, V.; Morcos, A.; Shekhar, S.; Goldstein, T.; Bordes, F.; Bardes, A.; Mialon, G.; Tian, Y.; Schwarzschild, A.; Wilson, A. G.; Geiping, J.; Garrido, Q.; Fernandez, P.; Bar, A.; Pirsiavash, H.; LeCun, Y.; and Goldblum, M. 2023. A Cookbook of Self-Supervised Learning. arXiv:2304.12210.

Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. 1986. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, 273–282.

Bruce, J.; Dennis, M.; Edwards, A.; Parker-Holder, J.; Shi, Y. J.; Hughes, E.; Lai, M.; Mavalankar, A.; Steigerwald, R.; Apps, C.; Aytar, Y.; Bechtle, S.; Behbahani, F.; Chan, S.; Heess, N.; Gonzalez, L.; Osindero, S.; Ozair, S.; Reed, S.; Zhang, J.; Zolna, K.; Clune, J.; De Freitas, N.; Singh, S.; and Rocktäschel, T. 2024. Genie: generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Countdown. 2025. Countdown (game show). [Online; accessed 12-October-2025].

Craik, K. J. W. 1967. *The nature of explanation*, volume 445. CUP Archive.

de Sa, V. 1993. Learning Classification with Unlabeled Data. In Cowan, J.; Tesauro, G.; and Alspector, J., eds., *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.

Dziri, N.; Lu, X.; Sclar, M.; Li, X. L.; Jiang, L.; Lin, B. Y.; Welleck, S.; West, P.; Bhagavatula, C.; Bras, R. L.; Hwang, J. D.; Sanyal, S.; Ren, X.; Ettinger, A.; Harchaoui, Z.; and Choi, Y. 2023. Faith and Fate: Limits of Transformers on Compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Edelman, E.; Tsilivis, N.; Edelman, B. L.; Malach, E.; and Goel, S. 2024. The Evolution of Statistical Induction Heads: In-Context Learning Markov Chains. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 64273–64311. Curran Associates, Inc.

Eldan, R.; and Li, Y. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Friston, K. 2010. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2): 127–138.

Gadre, S. Y.; Smyrnis, G.; Shankar, V.; Gururangan, S.; Wortsman, M.; Shao, R.; Mercat, J.; Fang, A.; Li, J.; Keh, S.; Xin, R.; Nezhurina, M.; Vasiljevic, I.; Jitsev, J.; Soldaini, L.; Dimakis, A. G.; Ilharco, G.; Koh, P. W.; Song, S.; Kollar, T.; Carmon, Y.; Dave, A.; Heckel, R.; Muennighoff, N.; and Schmidt, L. 2024. Language models scale reliably with over-training and on downstream tasks. arXiv:2403.08540.

Gandhi, K.; Lee, D.; Grand, G.; Liu, M.; Cheng, W.; Sharma, A.; and Goodman, N. D. 2024. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2170–2179. PMLR.

Gloeckle, F.; Idrissi, B. Y.; Rozière, B.; Lopez-Paz, D.; and Synnaeve, G. 2024. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Goyal, S.; Ji, Z.; Rawat, A. S.; Menon, A. K.; Kumar, S.; and Nagarajan, V. 2023. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*.

Gu, A.; and Dao, T. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752.

Gu, A.; Goel, K.; and Ré, C. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. arXiv:2111.00396.

Gurnee, W.; and Tegmark, M. 2024. Language Models Represent Space and Time. In *The Twelfth International Conference on Learning Representations*.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*, 2(3).

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Hafner, D.; Lillicrap, T. P.; Norouzi, M.; and Ba, J. 2021. Mastering Atari with Discrete World Models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2025. Mastering Diverse Control Tasks through World Models. 640(8059): 647–653.

Hansen, N.; Wang, X.; and Su, H. 2022. Temporal Difference Learning for Model Predictive Control. arXiv:2203.04955.

Hendrycks, D.; and Gimpel, K. 2023. Gaussian Error Linear Units (GELUs). arXiv:1606.08415.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hu, E. S.; Ahn, K.; Liu, Q.; Xu, H.; Tomar, M.; Langford, A.; Teoh, J.; Xu, B.; Yan, D.; Jayaraman, D.; Lamb, A.; and Langford, J. 2025. The Belief State Transformer. arXiv:2410.23506.

Huang, H.; LeCun, Y.; and Balestriero, R. 2025. LLM-JEPA: Large Language Models Meet Joint Embedding Predictive Architectures. *arXiv preprint arXiv:2509.14252*.

Johnson-Laird, P. N. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. 6. Harvard University Press.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Langford, J.; and Schapire, R. 2005. Tutorial on practical prediction theory for classification. *Journal of machine learning research*, 6(3).

Li, K.; Hopkins, A. K.; Bau, D.; Viégas, F.; Pfister, H.; and Wattenberg, M. 2023. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for MDPs. *AI&M*, 1(2): 3.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, B.; Ash, J. T.; Goel, S.; Krishnamurthy, A.; and Zhang, C. 2023. Transformers Learn Shortcuts to Automata. In *The Eleventh International Conference on Learning Representations*.

Liu, S.; Mallol-Ragolta, A.; Parada-Cabeleiro, E.; Qian, K.; Jing, X.; Kathan, A.; Hu, B.; and Schuller, B. W. 2022. Audio Self-supervised Learning: A Survey. arXiv:2203.01205.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Makkuva, A. V.; Bondaschi, M.; Girish, A.; Nagle, A.; Jaggi, M.; Kim, H.; and Gastpar, M. 2025. Attention with Markov: A Curious Case of Single-layer Transformers. In *The Thirteenth International Conference on Learning Representations*.

Merrill, W.; Petty, J.; and Sabharwal, A. 2025. The Illusion of State in State-Space Models. arXiv:2404.08819.

Merrill, W.; Sabharwal, A.; and Smith, N. A. 2022. Saturated Transformers are Constant-Depth Threshold Circuits. *Transactions of the Association for Computational Linguistics*, 10: 843–856.

Miall, R. C.; and Wolpert, D. M. 1996. Forward models for physiological motor control. *Neural networks*, 9(8): 1265–1279.

Nagarajan, V.; Wu, C. H.; Ding, C.; and Raghunathan, A. 2025. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. *arXiv preprint arXiv:2504.15266*.

Ni, T.; Eysenbach, B.; SeyedSalehi, E.; Ma, M.; Gehring, C.; Mahajan, A.; and Bacon, P.-L. 2024. Bridging State and History Representations: Understanding Self-Predictive RL. In *The Twelfth International Conference on Learning Representations*.

Patel, R.; and Pavlick, E. 2022. Mapping language models to grounded conceptual spaces. In *International conference on learning representations*.

Power, A.; Burda, Y.; Edwards, H.; Babuschkin, I.; and Misra, V. 2022. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. arXiv:2201.02177.

Roy, O.; and Vetterli, M. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, 606–610. IEEE.

Schmidhuber, J. 1990. *Making the World Differentiable: On Using Self Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments*. Forschungsberichte Künstliche Intelligenz. Inst. für Informatik.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Schwarzer, M.; Anand, A.; Goel, R.; Hjelm, R. D.; Courville, A.; and Bachman, P. 2021. Data-Efficient Reinforcement Learning with Self-Predictive Representations. arXiv:2007.05929.

Striebel, C. 1965. Sufficient statistics in the optimum control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12(3): 576–592.

Subramanian, J.; Sinha, A.; Seraj, R.; and Mahajan, A. 2022. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12): 1–83.

Sutton, R. S. 1991. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4): 160–163.

Tang, Y.; Guo, Z. D.; Richemond, P. H.; Pires, B. A.; Chandak, Y.; Munos, R.; Rowland, M.; Azar, M. G.; Lan, C. L.; Lyle, C.; György, A.; Thakoor, S.; Dabney, W.; Piot, B.; Calandriello, D.; and Valko, M. 2023. Understanding self-predictive learning for reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Thrush, T.; Potts, C.; and Hashimoto, T. 2025. Improving Pretraining Data Using Perplexity Correlations. arXiv:2409.05816.

Vafa, K.; Chang, P. G.; Rambachan, A.; and Mullainathan, S. 2025. What Has a Foundation Model Found? Using Inductive Bias to Probe for World Models. In *International Conference on Machine Learning*, 60727–60747. PMLR.

Vafa, K.; Chen, J. Y.; Rambachan, A.; Kleinberg, J.; and Mullainathan, S. 2024. Evaluating the world model implicit in a generative model. *Advances in Neural Information Processing Systems*, 37: 26941–26975.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wu, Z.; Qiu, L.; Ross, A.; Akyürek, E.; Chen, B.; Wang, B.; Kim, N.; Andreas, J.; and Kim, Y. 2024. Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 1819–1862. Mexico City, Mexico: Association for Computational Linguistics.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 11809–11822. Curran Associates, Inc.

Ye, J.; Gao, J.; Gong, S.; Zheng, L.; Jiang, X.; Li, Z.; and Kong, L. 2025. Beyond Autoregression: Discrete Diffusion for Complex Reasoning and Planning. In *The Thirteenth International Conference on Learning Representations*.

Ye, W.; Liu, S.; Kurutach, T.; Abbeel, P.; and Gao, Y. 2021. Mastering atari games with limited data. *Advances in neural information processing systems*, 34: 25476–25488.

Zhang, A.; McAllister, R.; Calandra, R.; Gal, Y.; and Levine, S. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*.

Zhang, K.; Wen, Q.; Zhang, C.; Cai, R.; Jin, M.; Liu, Y.; Zhang, J.; Liang, Y.; Pang, G.; Song, D.; and Pan, S. 2024. Self-Supervised Learning for Time Series Analysis: Taxonomy, Progress, and Prospects. arXiv:2306.10125.

A Belief States in Sequence Modeling

Recent work has introduced variants of sequence modeling architectures based on the principle of learning belief states, i.e., BST and JTP. We review these methods here. Let θ denote the parameters of a transformer-based model. Let $\mathbf{h}_{s:t}$ denote the hidden states produced by the transformer encoder for a token sequence $X_{s:t}$, where $s \leq t$. When we use the notation $\mathbf{h}_t$, it is shorthand for $\mathbf{h}_{1:t}$. The model's output head produces a categorical distribution over the token vocabulary conditioned on some hidden state representation, i.e., $p_\theta(\cdot \mid \mathbf{h}_{s:t})$.

Belief State Transformer. The Belief State Transformer (BST; Hu et al. (2025)) learns *compact* belief states by jointly training a next-token predictor and a previous-token predictor across all possible prefix–suffix decompositions of a sequence, including cases where either the prefix or suffix is empty. Concretely, given a prefix $X_{1:t}$ and a suffix $X_{t+k:T}$ with $k \geq 1$, BST aims to minimize the cross-entropy loss

$$\mathcal{L}_{\text{BST}}(\theta) = \mathbb{E}_{t<T}\Big[-\log \underbrace{p_\theta(X_{t+1} \mid \mathbf{h}_{1:t}, \mathbf{h}_{t+k:T})}_{\text{next-token prediction}} - \log \underbrace{p_\theta(X_{t+k-1} \mid \mathbf{h}_{1:t}, \mathbf{h}_{t+k:T})}_{\text{previous-token prediction}} \Big], \tag{6}$$

where $\mathbf{h}_{1:t}$ and $\mathbf{h}_{t+k:T}$ are produced by separate transformers. This bidirectional training shapes the hidden representations of the BST into belief states.

Joint Multi-token Prediction. Joint multi-token prediction (JTP; Ahn, Lamb, and Langford (2025)) aims to learn the *joint* distribution over the next $d+1$ tokens conditioned on $\mathbf{h}_t$, where d is the multi-step prediction horizon beyond the next token. Specifically, JTP minimizes the loss

$$\mathcal{L}_{\text{JTP}}(\theta; d, \lambda_{\text{JTP}}) = \mathbb{E}_{t<T}\Big[-\log \underbrace{p_\theta(X_{t+1} \mid \mathbf{h}_t)}_{\text{next-token prediction}} - \cdot \frac{1}{d}\sum_{i=1}^{d} \log \underbrace{p_\theta(X_{t+i+1} \mid \text{Fetch}(\mathbf{h}_t, X_{t+1:t+i}))}_{\text{joint multi-token prediction}} \Big], \tag{7}$$

where an additional module $\text{Fetch}(\mathbf{h}_t, X_{t+1:t+i})$ is used to create an embedding combining the teacher-forced tokens $X_{t+1:t+i}$ with $\mathbf{h}_t$. Although Ahn, Lamb, and Langford (2025) suggest that their method learn "short-horizon belief states", they do not formally define the conditions under which this occurs. To understand how JTP learns belief states, we start by defining a *k-observable* system.

Definition A.1 (k-observability for sequences)**.** A system is k-observable if for any two sequences $H = X_{1:t}$ and $H' = X_{1:j}$ that induce the same joint distribution over the next-k tokens, i.e., $\mathbb{P}(X_{t+1:t+k} \mid H) = \mathbb{P}(X_{t+1:t+k} \mid H')$, it follows that their *full-horizon* conditionals match:

$$\mathbb{P}(X_{t+1:T} \mid H) = \mathbb{P}(X_{t+1:T} \mid H'). \tag{8}$$

In other words, the distribution of all future observations is determined by the distribution of the next k observations.

Proposition A.2 (JTP forms belief states in k-observable systems)**.** *Assume the system is k-observable and let $k = d+1$. Suppose the joint multi-token prediction model recovers the true joint next-k conditional, i.e. $p_\theta(X_{t+1} \mid \mathbf{h}_t)p_\theta(X_{t+2} \mid \mathbf{h}_t, X_{t+1}) \dots p_\theta(X_{t+k} \mid \mathbf{h}_t, X_{t+1:t+k-1}) = \mathbb{P}(X_{t+1:t+k} \mid X_{1:t})$ a.s. for all t, then $\mathbf{h}_t$ is a belief state for $X_{1:t}$.*

Proof. By k-observability (Theorem A.1), there exists a measurable map G taking the joint next-k conditional distribution to the full-horizon conditional:

$$\mathbb{P}(X_{t+1:T} \mid X_{1:t}) = G\Big(\mathbb{P}(X_{t+1:t+k} \mid X_{1:t})\Big).$$

By the premise that JTP recovers the true next-k joint, conditioning on $\mathbf{h}_T$ is equivalent to conditioning on $X_{1:t}$ for all bounded measurable functionals of the future. Hence $\mathbf{h}_t$ is a belief state for all $t < T$. □

Intuitively, if all possible futures can be distinguished by the next k tokens, then a JTP model that accurately predicts the joint next-k distribution would encode all information necessary to distinguish future trajectories. Note that both next-token prediction and multi-token prediction do not guarantee belief state representations (c.f. Hu et al. (2025)).

B Formal Proof of Theorem 3.2

The proof follows the intuition illustrated below. Optimizing for next-token and transition consistency (Equations (1) and (2)) ensures the existence of measurable maps p_θ and p_ψ that allow recursive decoding of future tokens:

$$\mathbf{h}_t \xrightarrow[\text{decode token}]{p_\theta} X_{t+1} \xrightarrow[\text{update state}]{p_\psi} \mathbf{h}_{t+1} \xrightarrow[\text{decode token}]{p_\theta} X_{t+2} \xrightarrow[\text{update state}]{p_\psi} \mathbf{h}_{t+2} \cdots \xrightarrow{p_\theta} X_T.$$

A formal proof proceeds by backward induction on t. For the base case $t = T-1$, the claim follows directly from Equation (1), since $\mathbf{h}_{T-1}$ suffices to predict X_T.

Now assume $\mathbf{h}_{k+1}$ is a belief state for $X_{1:k+1}$. We will show that $\mathbf{h}_k$ is also a belief state for $X_{1:k}$. From $\mathbf{h}_k$, one can generate

$$X_{k+1} \sim p_\theta(\cdot \mid \mathbf{h}_k),$$
$$\mathbf{h}_{k+1} \sim p_\psi(\cdot \mid \mathbf{h}_k, X_{k+1}).$$

By next-token and transition consistency (Equations (1) and (2)), we have

$$\begin{aligned}
\mathbb{P}(X_{k+1:T} \mid \mathbf{h}_k) &= \mathbb{P}(X_{k+2:T} \mid X_{k+1}, \mathbf{h}_k)\, \underbrace{\mathbb{P}(X_{k+1} \mid \mathbf{h}_k)}_{\text{Equation (1)}} \\
&= \mathbb{P}(X_{k+2:T} \mid X_{k+1}, \mathbf{h}_k)\, \mathbb{P}(X_{k+1} \mid X_{1:k}) \\
&= \left[\int \mathbb{P}(X_{k+2:T},\ \mathbf{h}_{k+1} \mid X_{k+1}, \mathbf{h}_k)\, d\mathbf{h}_{k+1}\right] \mathbb{P}(X_{k+1} \mid X_{1:k}) \\
&= \left[\int \underbrace{\mathbb{P}(X_{k+2:T} \mid \mathbf{h}_{k+1}, X_{k+1}, \mathbf{h}_k)\, \mathbb{P}(\mathbf{h}_{k+1} \mid X_{k+1}, \mathbf{h}_k)}_{\text{Equation (2)}}\, d\mathbf{h}_{k+1}\right] \mathbb{P}(X_{k+1} \mid X_{1:k}) \\
&= \left[\int \underbrace{\mathbb{P}(X_{k+2:T} \mid \mathbf{h}_{k+1})}_{\text{Induction hypothesis}}\, \mathbb{P}(\mathbf{h}_{k+1} \mid X_{1:k+1})\, d\mathbf{h}_{k+1}\right] \mathbb{P}(X_{k+1} \mid X_{1:k}) \\
&= \mathbb{P}(X_{k+2:T} \mid X_{1:k+1})\, \mathbb{P}(X_{k+1} \mid X_{1:k}) = \mathbb{P}(X_{k+1:T} \mid X_{1:k}).
\end{aligned}$$

This proves that $\mathbf{h}_k$ is also a belief-state. This concludes the proof.

C More Details on NextLat Implementation

We parameterize the latent transition model p_ψ with a three-layer MLP using GELU (Hendrycks and Gimpel 2023) activations. The latent transition model takes as input the layer-normalized (Ba, Kiros, and Hinton 2016) concatenation of the current hidden state $\mathbf{h}_t$ and next-token embedding X_{t+1}, and outputs a *delta* update applied via residual connection:

$$\hat{\mathbf{h}}_{t+1} = p_\psi(\mathbf{h}_t, X_{t+1}) = f_\psi(\mathbf{h}_t, X_{t+1}) + \mathbf{h}_t \tag{9}$$

where $f_\psi(\cdot)$ predicts the modification to $\mathbf{h}_t$ (see Figure 5). This paper aims to demonstrates that NextLat yields significant performance gains even with a **simple MLP latent transition**. We foresee even better performances with more sophiscated latent transition model architectures, but we leave that exploration to future work.

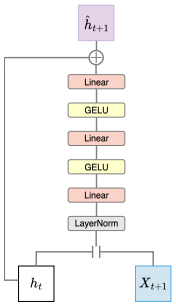


Figure 5: Illustration of the latent transition model p_ψ.

D Experiment Details

In this section, we provide more details on the setup of the experiments in the main body. Our experiments are executed on NVIDIA H100 NVL GPUs and most of them complete within 48 hours on a single GPU, except those involving MTP and BST which are more computationally intensive. Table 4 provides an overview of the training, model, and NextLat hyperparameters across all experimental domains.

	Manhattan	**Countdown**	**Path-Star**	**TinyStories**
Training Parameters				
Steps	400k (6 epochs)	100k (204 epochs)	20k (51 epochs)	100k (8.5 epochs)
Batch Size	256	1024	512	256
Learning Rate	1e-4	3e-4	5e-4	3e-4
Weight Decay	0.01	0.1	0.1	0.1
Optimizer	*AdamW* (Loshchilov and Hutter 2019) with $\beta_1 = 0.9,\ \beta_2 = 0.95$			
Model Parameters				
Layers	48	12	12	8
Heads	8	12	6	8
Hidden Dimension	384	768	384	768
NextLat Parameters				
$\lambda_{\text{next-h}}$	1.0	2.0	1.0	1.0
λ_{KL}	0.1	1.0	1.0	1.0
p_ψ MLP Hidden Dimension	1536	768	384	2048
p_ψ MLP Layers	3	3	3	3

Table 4: Training, Model, and NextLat hyperparameters across all benchmarks.

Algorithm 1: Pseudocode for NextLat's loss function in PyTorch syntax. The loss functions `cross_entropy_loss()`, `smooth_l1_loss()`, and `KL_loss()` are implemented externally.

```
import torch.nn as nn
from copy import deepcopy

def loss(batch, targets):
    # batch:(B,T), targets: (B,T)
    # embedding dimension = D, vocabulary size = V
    batch_size, seq_len = batch.shape

    # Encode sequences into latent states
    hidden_states = Transformer(batch) # (B,T,D)

    # Compute next-token loss
    logits_post = Output_Head(hidden_states) # (B,T,V)
    loss_next = cross_entropy_loss(logits_post, targets)

    initial_hidden = torch.zeros(batch_size, 1, 1) # (B,1,1)
    next_tokens = batch
    next_states = hidden_states
    current_states = hidden_states
    loss_next_h = 0
    loss_kl = 0

    # Recursive multi-step predictions
    for _ in range(multi_step_horizon):
        # Shift hidden states back by 1 using dummy initial state, similar to RNNs
        current_states = torch.cat([initial_hidden, current_states[:, :-1]], dim=1)
        # Predict next hidden state using latent transition model
        pred_next_states = Latent_Transition(current_states, next_tokens) #(B,T,D)
        # Compute next-hidden loss using detached next states as targets
        loss_kl += smooth_l1_loss(pred_next_states, next_states.detach())

        # Compute KL loss using detached output head
        logits_prior = deepcopy(Output_Head)(pred_next_states) # (B,T,V)
        loss_kl += KL_loss(logits_post.detach(), logits_prior) # detach posterior

        current_states = pred_next_states

    loss_next_h = loss_next_h / belief_interval
    loss_kl = loss_kl / belief_interval
    # overall NextLat loss
    return loss_next + next_h_lambda * loss_next_h + kl_lambda * loss_kl
```

Due to computational constraints, we did not perform exhaustive hyperparameter tuning for NextLat. Instead, the hyperparameters reported in Table 4 were chosen through a small-scale search, exploring $\lambda_{\text{next-h}} \in \{1.0, 2.0\}$ and $\lambda_{\text{KL}} \in \{0.1, 1.0\}$, guided by empirical observations and intuition. Encouragingly, we find that NextLat performs robustly over a wide range of settings. In particular, λ_{KL} requires minimal tuning; $\lambda_{\text{KL}} = 1.0$ works decently across all tasks. After all, it primarily serves as a complementary alignment objective. Likewise, $\lambda_{\text{next-h}} = 1.0$ is effective in most cases, though slightly higher values (e.g., $\lambda_{\text{next-h}} = 2.0$) was beneficial when the next-latent regression loss is of much smaller scale than the token-level losses (i.e., $\mathcal{L}_{\text{next-token}}$ and $\mathcal{L}_{\text{KL}}$).

Manhattan Taxi Rides

Here, we provide additional details on our training and evaluation setups for the Manhattan Taxi Rides benchmark and clarify key differences from the original study of Vafa et al. (2024).

Training. Since this task inherently requires state tracking (i.e., tracking position within Manhattan), and increasing model depth is known to benefit transformers on such tasks (Merrill, Petty, and Sabharwal 2025), we employ 48-layer transformers with 384 hidden dimensions and 8 attention heads (89M parameters). This differs from Vafa et al. (2024), which used 12 layers, 768 hidden dimensions, and 12 heads for their smaller transformer variant. We found that increasing model depth yielded substantial performance gains, whereas increasing hidden dimensionality offered negligible improvement. As shown in Table 1, the effective latent rank of our models is substantially smaller than 384, suggesting that large hidden dimensions are unnecessary.

Unlike the original study, which trained models for only one epoch, our models are trained for six epochs, as we observed that performances generally do not converge within a single epoch. This also helps rule out potential "grokking" phenomena (Power et al. 2022), where generalization improves only after extended periods of overfitting. To enable longer training without substantially increasing runtime, we apply sequence packing, i.e., concatenating multiple sequences into longer ones while masking cross-sequence attention. This enables efficient utilization of GPU memory and computation. Most models complete six training epochs in under three days on a single NVIDIA H100 NVL GPU, except MTP, which has substantially more parameters than the others.

Evaluation. Models are trained on random traversals of length 100 connected pickup and dropoff intersections. This task inherently requires not only state tracking but also planning, as models must reason over possible future paths to generate valid 100-step trajectories that reach the destination while avoiding dead ends, i.e., road segments disconnected from the goal due to the one-way streets. Random traversals rarely correspond to true shortest paths and do not provide an inductive bias toward learning the shortest-path algorithm, which relies on dynamic programming. Consequently, unlike Vafa et al. (2024), who evaluated pairs with shortest paths of up to 100 steps, we limit evaluation pairs to paths of up to 50 steps. This adjustment ensures that evaluation pairs do not demand long-horizon planning beyond the training distribution, which might otherwise force the model to produce forced predictions to compensate for planning failure. This also ensures that inconsistencies in a model's internal map are more reflective of world-model incoherence rather than artifacts of long-horizon planning limitations. These evaluation pairs are used to generate **??** and to compute the sequence compression and detour robustness metrics, following the procedure described in Vafa et al. (2024).

For the effective latent rank, we pass a batch of 256 sequences (each of length 256) through the model to obtain the hidden state matrix. Singular values smaller than $1\text{e}{-12}$ are discarded, and the effective rank is then computed following Roy and Vetterli (2007). For GPT and NextLat, we use the final-layer hidden states. For JTP, we extract the hidden states immediately before the self-attention module in the Fetch head (see Equations 4–5 in Ahn, Lamb, and Langford (2025)). For MTP, we use the output of the next-token prediction head to compute the effective rank.

Countdown

We largely follow Gandhi et al. (2024) for the Countdown training and evaluation setup. Each problem consists of four input numbers and a solution sequence comprising three equations, consistent with prior work (Gandhi et al. 2024; Ye et al. 2025). A training example is formatted as

$$14, 83, 88, 91, 23 \mid 83 - 14 = 69,\ 91 - 88 = 3,\ 69/3 = 23$$

where the first four numbers are the inputs, the fifth is the target number, and the pipe symbol "$\mid$" separates the input prompt from the solution. During training, loss values corresponding to input prompt are masked out.

Previous studies involving the Countdown benchmark used pretrained GPT-2 byte-pair encoding tokenizers, which do not necessarily tokenize multi-digit numbers as single units. In contrast, we construct a custom tokenizer that assigns each integer from 1 to 10,000 to a unique token, ensuring that every number in the sequence is represented atomically. The arithmetic operators and delimiters, i.e., $\{\ \mid\ ,\ +\ ,\ -\ ,\ \times\ ,\ \div\ \}$, are each assigned their own token indices. Due to the large branching factor of the Countdown problem, we insert eight pipe symbols ("$\mid$") between the input and the solution as pause tokens (Goyal et al. 2023), allowing the model additional computation steps to plan before generating its answer.

Path-Star

Our Path-Star data preparation, training, and evaluation follow Bachmann and Nagarajan (2024), except that we increase the weight decay to 0.1, which we found helpful for stable convergence and higher solve rates in the multi-step prediction methods (i.e., MTP, JTP, and NextLat). We evaluate each model's ability to generate the correct arm on 20k held-out test instances. Unlike Hu et al. (2025) and Ahn, Lamb, and Langford (2025) which generate a fresh set of graphs every batch, we adopt the original, more challenging setup of Bachmann and Nagarajan (2024), which uses a fixed sample size of 200k and node values sampled from $N = 100$. This difference accounts for the performance gap observed in the BST and JTP baselines in Figure 3b. The Path-Star experiment is designed to expose the myopic behavior of teacher-forced next-token prediction, which can encourage models to exploit superficial regularities—an effect referred to as the *Clever Hans cheat* (Bachmann and Nagarajan 2024). Because the task's sample space grows exponentially with graph size, identifying the correct algorithm that generalizes across all graph instances is highly nontrivial. While not conclusive, our results suggest that latent-space prediction and the inductive bias toward compressing history into belief states promote better discovery of generalizable solutions in data-constrained settings.

TinyStories

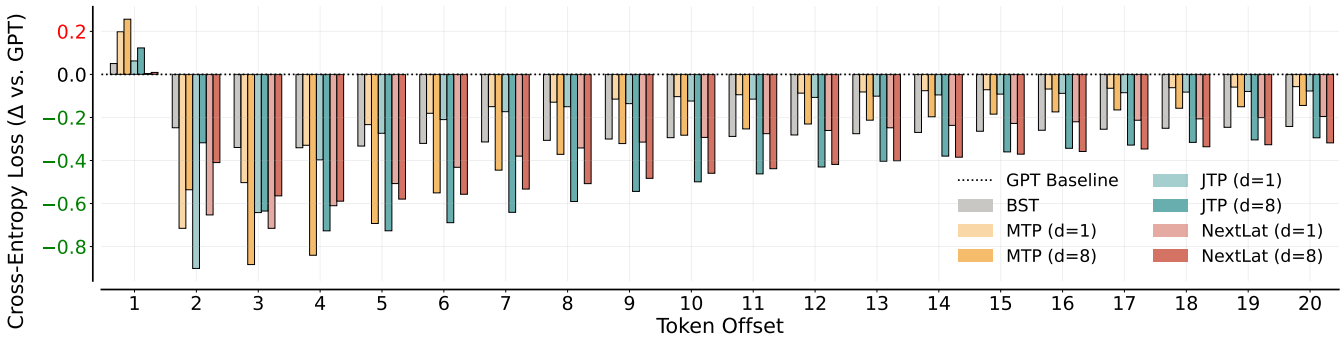


Figure 6: Full plot version of Figure 4 that shows probe performance across all 20 tokens offsets.

Our TinyStories setup follows exactly that of Hu et al. (2025). After pretraining on TinyStories, we train linear (one-layer) probes on the hidden states of the frozen transformer models for an additional 20k steps on the same dataset. All probe training hyperparameters (e.g., learning rate, batch size) match those used during pretraining (see Table 4).

For GPT, JTP, and NextLat, the choice of hidden states follows the setup used for measuring effective rank in the Manhattan taxi rides task, as described in Section D. For BST, we use the final-layer hidden states of the forward transformer encoder. For MTP (Gloeckle et al. 2024), we use the output of the shared transformer trunk, i.e., the hidden state before it branches into separate transformer heads for multi-token prediction, as this final shared representation contains the most predictive information about future tokens.

E Future Work

We are excited to extend this preliminary study of NextLat to a broader set of experiments. In future work, we plan to evaluate NextLat in large-scale language model pretraining, investigating whether its self-predictive latent objective can improve representation quality, data efficiency, and downstream performance in open-ended text modeling. We also intend to explore NextLat as a finetuning objective. Because it introduces no architectural modifications to the transformer, NextLat can be applied directly on top of existing pretrained models. This makes it a promising candidate for post-hoc finetuning, potentially improving reasoning, planning, or world-modeling capabilities in pretrained language models for downstream tasks without retraining from scratch. Finally, we aim to investigate architectural extensions that may further enhance NextLat's effectiveness. For example, we plan to experiment with higher-dimensional or hierarchical belief states obtained by combining hidden representations across multiple layers or temporal contexts. Such designs could enable richer latent transition dynamics and improve the model's expressivity to perform structured reasoning and planning over longer horizons.