

Physics-Informed Trajectory Constraints in Transformer Residual Streams

Tristyn Meaux
Independent Researcher, Austin, TX
tristynmeaux@yahoo.com

April 2026

Abstract

We investigate whether physics-informed trajectory constraints—soft penalties borrowed from the PINN framework—can be imposed on a transformer language model’s residual stream during training without degrading language modeling performance. We attach a lightweight scalar readout head to the residual stream and penalize trajectories that violate specified dynamical laws: monotonicity (the readout can only increase) and exponential decay (the readout fades between events). Across 66 training runs spanning two datasets (TinyStories, WikiText-103), two model scales (10.8M, 25M parameters), and four conditions (vanilla, multitask, monotone, decay), we find that trajectory constraints cost 0–1.9% perplexity, with the cost shrinking along both scale and training duration. Constrained models exhibit expanded representational dimensionality (+20–37% PCA participation ratio), achieve target recoverability $r^2 > 0.90$ across constraint types, and—most strikingly—preserve features that unconstrained training spontaneously develops and then degrades. Our results establish PINN-style trajectory constraints as a viable, low-cost mechanism for imposing structural properties on transformer representations at training time.

1 Introduction

Large language models develop rich internal representations during training, encoding far more than next-token prediction demands. Recent interpretability work has revealed structured, high-level features within transformer residual streams—from sentiment and truthfulness directions [Zou et al., 2023] to millions of monosemantic concepts [Templeton et al., 2024]. These findings demonstrate that transformers naturally organize their internal geometry around interpretable axes.

However, these features are *discovered*, not designed. They emerge unpredictably, vary across training runs, and—as we show in this work—can be actively degraded by continued training. Current approaches to leveraging internal structure operate post-hoc: representation engineering [Zou et al., 2023] steers activations at inference time, while sparse autoencoders [Templeton et al., 2024] decompose trained representations into interpretable components. Neither approach provides structural guarantees about what a model’s residual stream will contain after training.

We ask a different question: **can we specify, at training time, what trajectory-shaped features a transformer’s residual stream must carry—and does this damage the model?**

We borrow from physics-informed neural networks (PINNs; Raissi et al. 2019), which encode physical laws as soft penalties in the training loss. In the PINN framework, a neural network learns to approximate a solution while respecting differential equations that govern the system’s behavior. We adapt this paradigm to language model training: rather than enforcing a PDE, we attach a

lightweight scalar readout head to the transformer’s residual stream and penalize trajectories that violate a specified law—monotonicity (the readout can only increase) or exponential decay (the readout fades at a fixed rate between events).

Our experiments yield six findings across two datasets (TinyStories, WikiText-103), two model scales (10.8M, 25M parameters), four training conditions (vanilla, multitask, monotone, decay), and three seeds per cell:

1. **PINN trajectory constraints are compatible with language modeling.** Monotonicity costs 0.4–1.9% perplexity; decay costs nothing (and occasionally helps).
2. **The cost shrinks with both scale and training duration.** The monotonicity gap narrows from 0.75% to 0.44% between 10.8M and 25M parameters, and from 0.75% to 0.10% between 5,000 and 20,000 training steps on TinyStories.
3. **Constrained representations expand rather than compress.** PCA participation ratio increases 20–37% under monotonicity, while CKA between vanilla and constrained models remains stable at ~ 0.86 .
4. **The trajectory penalty contributes beyond multitask learning.** Models trained with readout prediction alone achieve high M recoverability ($r^2 > 0.92$) but their violation rates remain at chance.
5. **The approach generalizes across constraint types.** Monotonicity and decay—two structurally distinct trajectory laws—are both learned with comparable fidelity ($r^2 = 0.90$ – 0.97).
6. **PINN constraints preserve features that unconstrained training degrades.** Vanilla models spontaneously develop M -correlated representations early in training ($r^2 \approx 0.74$ at step 2,000), then compress them away ($r^2 \approx 0.53$ at step 20,000). Constrained models hold steady at $r^2 \approx 0.96$ throughout.

2 Related Work

Residual stream interpretability. Anthropic’s work on scaling monosemanticity [Templeton et al., 2024] demonstrated that sparse autoencoders can extract millions of interpretable features from production language models, revealing rich internal structure including safety-relevant concepts. Our work complements this by showing that specific features can be *imposed* during training rather than discovered post-hoc, and that training-time constraints preserve features that would otherwise degrade.

Representation engineering. Zou et al. [2023] introduced methods for monitoring and manipulating high-level representations in LLMs, achieving state-of-the-art honesty control by steering activation vectors at inference time. Their approach intervenes at inference; ours intervenes at training. The two are complementary—representation engineering identifies *which* features matter; PINN constraints could ensure those features remain structurally present after training.

Auxiliary prediction in transformers. Next-Latent Prediction [Sinha et al., 2025] trains transformers to predict their own future hidden states, encouraging convergence toward belief states without changing the architecture. Belief State Transformers [Hu et al., 2024] take a dual-encoder approach to similar ends. Both add auxiliary prediction objectives to the residual stream. Our

multitask condition is analogous—and our ablation (Section 4.4) demonstrates that prediction alone does not induce the trajectory property. The trajectory penalty is the active ingredient, not the auxiliary head.

Physics-informed neural networks. Raissi et al. [2019] introduced PINNs for solving differential equations by encoding physical laws as soft penalties in the training loss. The PINN paradigm has been extensively applied in computational physics but, to our knowledge, not to language model representation learning. We adopt the core idea—a penalty term that enforces a specified dynamical law—and apply it to the residual stream of a transformer, treating the readout trajectory as the “solution” that must respect the imposed “physics.”

Monotonic and constrained networks. Prior work on monotonic neural networks [Sill, 1997, Daniels and Velikova, 2010] enforces input-output monotonicity for calibration or fairness applications. Our monotonicity constraint is structurally different: it operates on the *internal trajectory* of a scalar readout across token positions within a sequence, not on the input-output mapping of the network.

3 Method

We investigate whether physics-inspired trajectory constraints can be imposed on a transformer language model’s internal representations during training without degrading language modeling performance. Our approach adds a lightweight scalar readout head to the transformer’s residual stream, trained jointly with the standard language modeling objective under a trajectory-specific penalty (Figure 1). We test two structurally distinct trajectory laws—monotonicity and decay—across two datasets and two model scales.

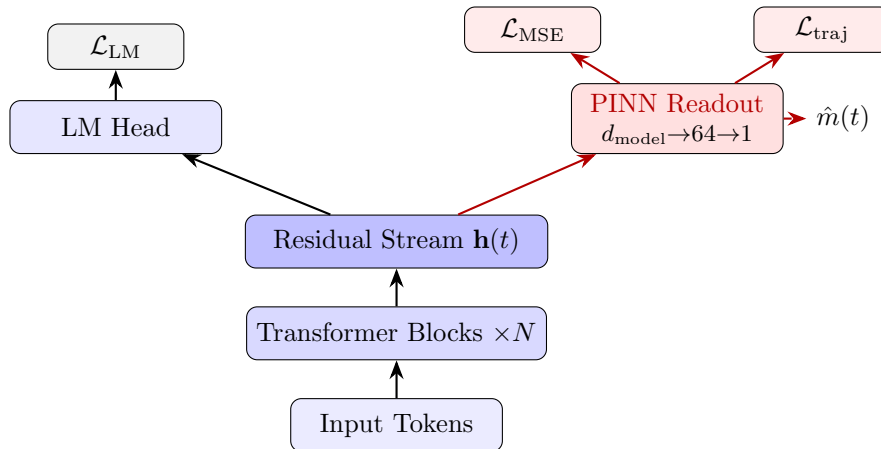


Figure 1: Architecture overview. A scalar PINN readout head (red) branches from the residual stream and is trained with both a prediction loss (\mathcal{L}_{MSE}) and a trajectory constraint ($\mathcal{L}_{\text{traj}}$), jointly with the standard language modeling objective (\mathcal{L}_{LM}). The readout head is present but ungraded in vanilla baselines.

3.1 Architecture

Our base model is a decoder-only transformer following the nanoGPT design [Karpathy, 2022], with causal self-attention, GELU activations, and tied input-output embeddings. We evaluate at two scales:

	10.8M	25M
Layers	6	8
Heads	6	8
Embedding dim	384	512
Head dim	64	64
MLP hidden dim	1536	2048
Block size	256	256

The head dimension ($d_{\text{head}} = 64$) is held constant across scales so that attention head dynamics remain comparable; only depth and width increase.

Readout head. We attach a two-layer MLP readout head to the final-layer residual stream hidden state $\mathbf{h}(t)$:

$$\hat{m}(t) = \mathbf{W}_2 \cdot \text{GELU}(\mathbf{W}_1 \cdot \mathbf{h}(t) + \mathbf{b}_1) + b_2 \quad (1)$$

where $\mathbf{W}_1 \in \mathbb{R}^{64 \times d_{\text{model}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{1 \times 64}$, producing a scalar prediction $\hat{m}(t)$ at every token position. The readout head is present in all experimental conditions, including vanilla baselines where it receives no gradient signal. This ensures that architectural differences do not confound perplexity comparisons.

3.2 Target Signal

We define $M(t)$, the ground-truth trajectory, as the normalized cumulative count of sentence-ending punctuation tokens ($.$, $?$, $!$) up to position t in each sequence:

$$M(t) = \frac{1}{Z} \sum_{i=1}^t \mathbb{1}[\text{token}_i \in \{., ?, !\}] \quad (2)$$

where Z is a per-dataset normalization constant equal to the mean count of integration tokens per sequence. $M(t)$ is monotonically non-decreasing by construction. For the decay constraint, we define a separate target:

$$M_{\text{decay}}(t) = \gamma \cdot M_{\text{decay}}(t-1) + \mathbb{1}[\text{token}_t \in \{., ?, !\}]/Z \quad (3)$$

with $\gamma = 0.95$, producing a leaky-cumulative signal that rises on sentence boundaries and exponentially decays between them.

3.3 Training Objectives

All models are trained with a standard character-level cross-entropy language modeling loss \mathcal{L}_{LM} . We define three experimental conditions by varying the auxiliary losses:

Vanilla. $\mathcal{L} = \mathcal{L}_{\text{LM}}$. The readout head exists but receives no gradient ($\lambda_{\text{readout}} = \lambda_{\text{constraint}} = 0$).

Multitask. $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{readout}} \cdot \mathcal{L}_{\text{MSE}}$, where $\mathcal{L}_{\text{MSE}} = \text{mean}((\hat{m}(t) - M(t))^2)$. The readout is trained to predict $M(t)$ but no trajectory constraint is imposed.

Constrained (monotone or decay). $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda_{\text{readout}} \cdot \mathcal{L}_{\text{MSE}} + \lambda_{\text{constraint}} \cdot \mathcal{L}_{\text{trajectory}}$, where $\mathcal{L}_{\text{trajectory}}$ takes one of two forms:

- *Monotonicity:* $\mathcal{L}_{\text{mono}} = \text{mean}(\text{ReLU}(\hat{m}(t) - \hat{m}(t+1)))$ — a one-sided penalty on any decrease in the readout between adjacent positions.
- *Decay:* $\mathcal{L}_{\text{decay}} = \text{mean}((\hat{m}(t+1) - \gamma \cdot \hat{m}(t) - s \cdot \text{integration}(t+1))^2)$ — a two-sided penalty enforcing the leaky-cumulative update law, where s is a normalization scale and $\gamma = 0.95$.

In all auxiliary conditions, $\lambda_{\text{readout}} = \lambda_{\text{constraint}} = 1.0$.

3.4 Datasets

TinyStories [Eldan and Li, 2023]. A corpus of short children’s stories generated by GPT-3.5/4, encoded at character level with a vocabulary of 128 tokens. Integration token density: $\sim 2.1\%$.

WikiText-103 [Merity et al., 2017]. Raw Wikipedia articles encoded at byte level (vocabulary size 256). Integration token density: $\sim 0.8\%$, roughly $3\times$ sparser than TinyStories.

3.5 Training Protocol

All models are trained with AdamW ($\beta_1=0.9$, $\beta_2=0.999$), learning rate 3×10^{-4} with 200-step linear warmup and cosine decay, weight decay 0.1, batch size 32, sequence length 256.

Matched-step experiments. Primary comparisons use 5,000 training steps across all conditions and seeds ($n=3$ per cell), corresponding to approximately 40M tokens. All conditions see identical data in identical order for each seed.

Chinchilla verification. To assess whether the perplexity gap stabilizes at longer training, we run three-seed experiments at 20,000 steps ($\sim 80\%$ Chinchilla-optimal for 10.8M), with intermediate checkpoints every 2,000 steps for probe analysis.

3.6 Evaluation Metrics

Language modeling. Validation perplexity on held-out data, reported as mean \pm standard deviation across seeds.

Representation analysis. We run post-hoc probes on each checkpoint: linear probes (Ridge regression) for $M(t)$, $M_{\text{decay}}(t)$, and character-type features; PCA participation ratio (effective dimensionality $= (\sum \lambda_i)^2 / \sum \lambda_i^2$); and linear Centered Kernel Alignment (CKA) between vanilla and constrained checkpoints on matched inputs.

4 Results

We organize our findings around six key results. All reported values are mean \pm standard deviation across $n=3$ seeds unless otherwise noted.

4.1 PINN Constraints Preserve Language Modeling Performance

Table 1: Validation perplexity across all conditions, datasets, and scales (5,000 steps, $n=3$). Percentages indicate relative gap versus vanilla.

	Vanilla	Multitask	Monotone	Decay
TS 10.8M	2.128 ± 0.008	2.135 (+0.3%)	2.144 (+0.8%)	2.126 (−0.1%)
TS 25M	2.049 ± 0.003	2.053 (+0.2%)	2.058 (+0.4%)	2.048 (−0.05%)
WT 10.8M	3.417 ± 0.007	3.507 (+2.6%)	3.480 (+1.9%)	3.391 (−0.8%)
WT 25M	3.236 ± 0.018	3.280 (+1.4%)	3.270 (+1.1%)	3.235 (−0.03%)

Figure 4: Perplexity across conditions, datasets, and scales

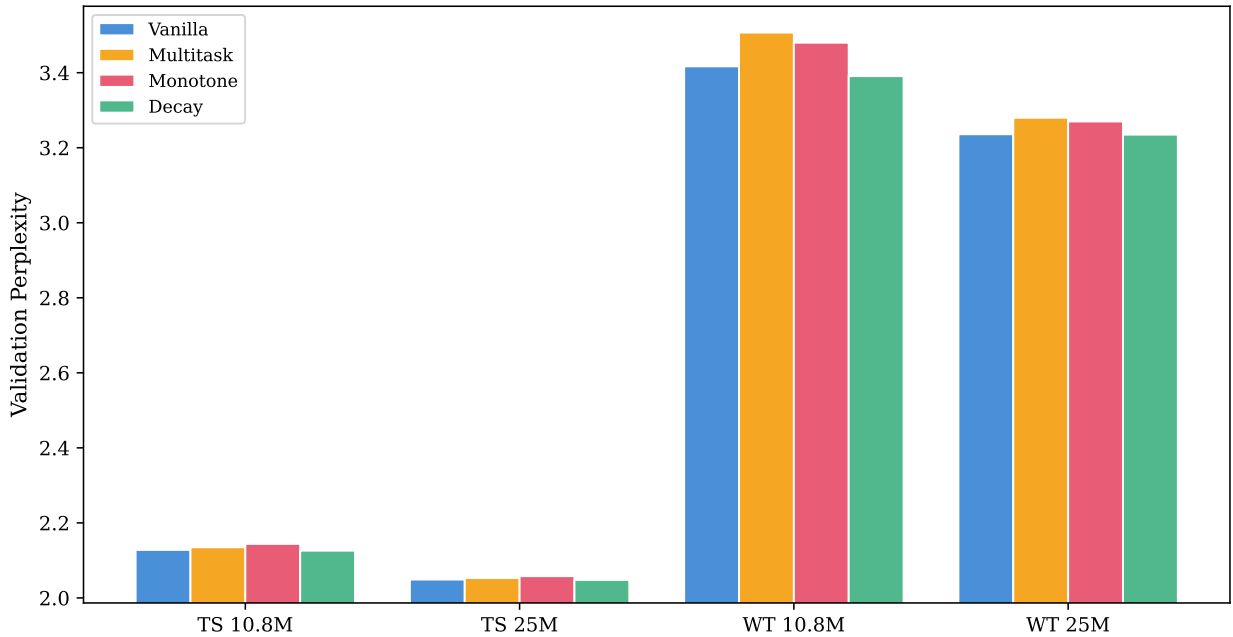


Figure 2: Validation perplexity across all conditions. Decay (green) consistently matches or improves upon vanilla (blue), while monotone (red) incurs a small positive cost. All gaps are below 2%.

The monotonicity constraint incurs a perplexity cost of 0.4–1.9% depending on dataset and scale. The decay constraint matches or slightly improves upon vanilla perplexity across all cells, suggesting that the leaky-cumulative trajectory provides useful inductive bias for language modeling.

4.2 Cost Shrinks with Scale and Training Duration

The perplexity gap decreases along two independent axes: model scale and training duration.

At full Chinchilla training on TinyStories, the monotonicity gap is 0.10%—within measurement noise. The 5,000-step costs in Table 1 are therefore conservative upper bounds on the true asymptotic cost.

Table 2: Chinchilla verification: perplexity gap at 5,000 vs. 20,000 steps ($n=3$).

Dataset	Vanilla (20K)	Monotone (20K)	Gap 5K	Gap 20K	Reduction
TinyStories	1.935 ± 0.009	1.937 ± 0.005	0.75%	0.10%	7.5 \times
WikiText	2.928 ± 0.009	2.956 ± 0.010	1.85%	0.96%	1.9 \times

Figure 1: Monotone perplexity gap shrinks with training

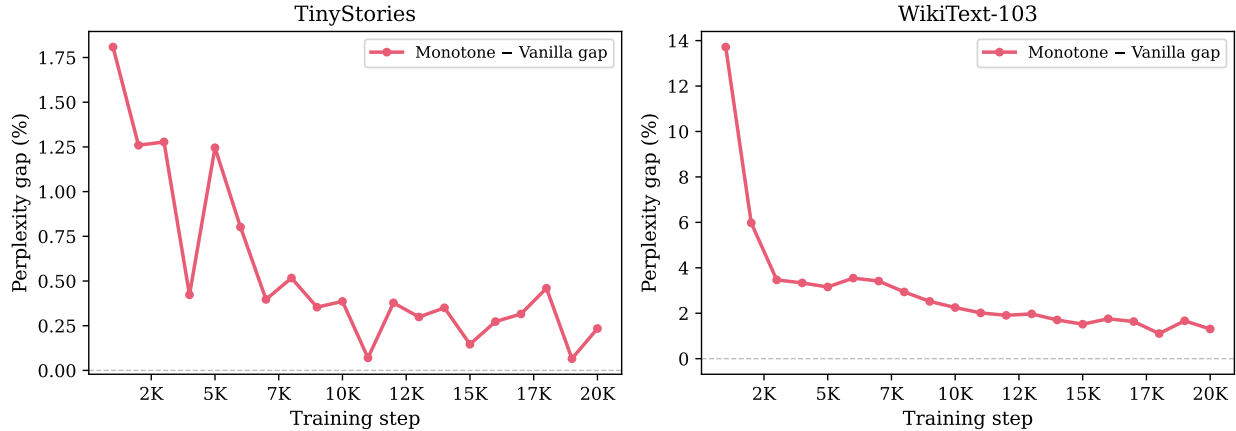


Figure 3: Monotone perplexity gap (%) as a function of training step. Both datasets show a clear downward trend, with TinyStories reaching near-zero by 20K steps.

4.3 PINN Constraints Expand Representations

Counter to the intuition that auxiliary constraints should compress representations, PINN-constrained models exhibit *higher* effective dimensionality.

Table 3: Representation geometry (seed 1 probes). PR = PCA participation ratio.

Cell	Vanilla PR	Monotone PR	Change	CKA
TS 10.8M	21.0	25.3	+20%	0.864
TS 25M	22.5	28.6	+27%	0.858
WT 10.8M	20.2	27.7	+37%	0.866
WT 25M	21.2	28.7	+35%	0.856

The participation ratio increases by 20–37% under monotonicity (Figure 4). CKA is stable at ~ 0.86 , indicating that approximately 86% of the representational geometry is preserved while the remaining 14% reorganizes to accommodate the trajectory constraint.

4.4 Trajectory Penalties Contribute Beyond Multitask Learning

Multitask models achieve high M recoverability ($r^2 > 0.92$) but their violation rates remain near chance (0.49–0.50). Monotone models achieve both high r^2 and reduced violation rates (0.40–0.47), demonstrating that the trajectory penalty induces a qualitatively different representational property.

Figure 3: PINN constraints expand representational dimensionality

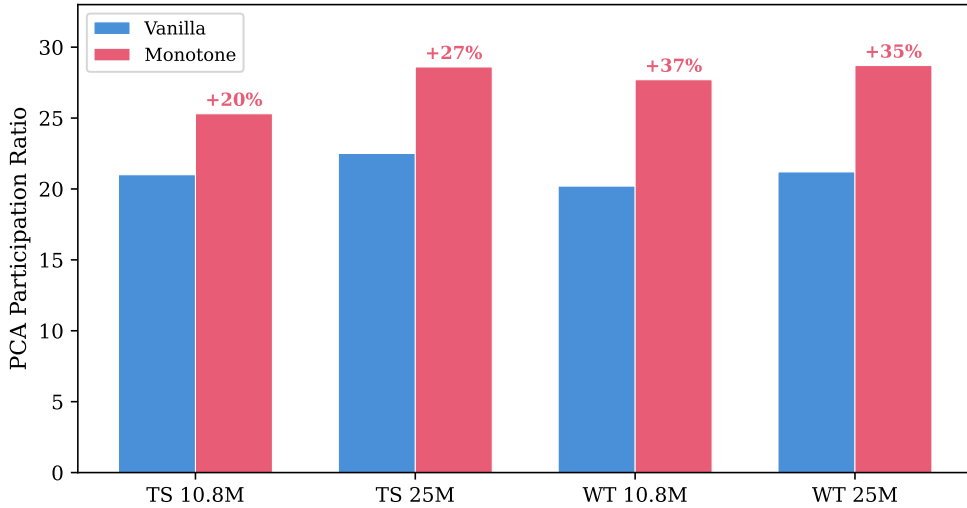


Figure 4: PCA participation ratio across conditions. Monotonicity increases effective dimensionality by 20–37% in every cell.

Table 4: Multitask vs. monotone: probe comparison (seed 1). Violation rate = fraction of adjacent positions where $\hat{m}(t) > \hat{m}(t+1)$.

Cell	Multitask r^2	Monotone r^2	MT viol.	Mono viol.
TS 10.8M	0.955	0.955	0.471	0.400
TS 25M	0.954	0.968	0.486	0.400
WT 10.8M	0.961	0.956	0.499	0.466
WT 25M	0.961	0.971	0.497	0.466

4.5 Constraint Diversity: Two Laws, Both Work

Each constraint successfully induces its own target ($r^2 = 0.90$ – 0.97) while cross-target probing shows partial but incomplete transfer ($r^2 = 0.53$ – 0.89). The decay constraint achieves this at zero perplexity cost (Table 1), making it strictly more efficient than monotonicity.

4.6 PINN Constraints Preserve Features That Vanilla Training Degrades

Our most unexpected finding: vanilla models spontaneously develop M -correlated structure early in training, then *lose* it as training progresses.

This reframes the role of PINN constraints: they do not merely *add* a feature to the representation. They *preserve* a feature that the model naturally develops and would otherwise abandon.

5 Discussion

Why does the decay constraint help? The decay target M_{decay} encodes a recency-weighted sentence boundary signal: recent sentence endings contribute more than distant ones, with exponential falloff at $\gamma = 0.95$. This is arguably more useful for next-character prediction than the cumulative count M —knowing that a sentence ended 5 tokens ago is more predictive of what comes

Table 5: Cross-constraint probing (r^2 for M and M_{decay} , seed 1). Bold = each constraint’s own target.

Cell	Decay $\rightarrow M_d$	Decay $\rightarrow M$	Mono $\rightarrow M$	Mono $\rightarrow M_d$
TS 10.8M	0.901	0.727	0.955	0.859
WT 10.8M	0.920	0.529	0.956	0.746
TS 25M	0.920	0.793	0.968	0.890
WT 25M	0.927	0.561	0.971	0.703

Figure 2: PINN constraints preserve features that vanilla training degrades

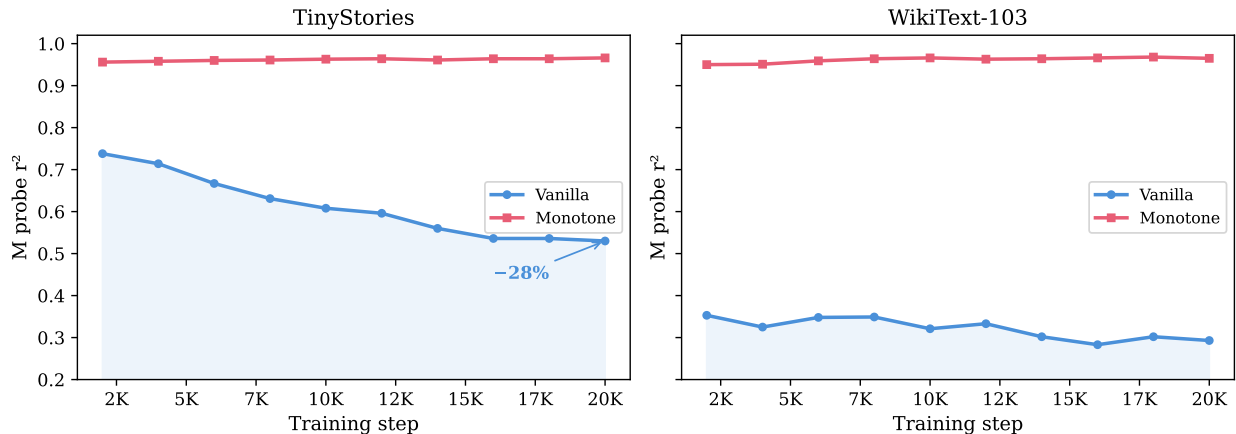


Figure 5: M probe r^2 as a function of training step. Vanilla models (blue) develop M -tracking early and then compress it away; monotone models (red) preserve it at $r^2 \approx 0.96$ throughout. On TinyStories, vanilla r^2 drops 28% over 20,000 steps.

next than knowing that 3 sentences have ended so far. The decay constraint thus provides an inductive bias aligned with the language modeling objective.

Why does multitask cost more than monotone on WikiText? A surprising result in Table 1 is that the multitask condition—which lacks a trajectory penalty—incurs a *larger* perplexity cost than monotonicity on WikiText (+2.6% vs. +1.9% at 10.8M). We hypothesize that the trajectory penalty acts as a regularizer: by constraining the readout to follow a specific trajectory, it reduces the effective number of free parameters the auxiliary head can use to “compete” with the language modeling objective for representational capacity. The unconstrained multitask head, by contrast, is free to overfit to M prediction in ways that are locally optimal for MSE but globally harmful to the residual stream’s utility for language modeling.

Why do representations expand? Standard language modeling concentrates representational capacity on features that reduce cross-entropy loss. Adding a trajectory constraint creates a secondary gradient signal that rewards distributing information across additional dimensions to simultaneously satisfy both objectives. The result is a higher participation ratio—the model uses more of its available representational capacity.

The fading phenomenon. Our most striking finding (Figure 5) is that vanilla models spontaneously develop M -correlated structure early in training and then degrade it. We interpret this as a consequence of representational competition: early in training, many features are useful for rapid loss reduction; as training progresses, the model converges toward a more efficient representation that discards features not directly needed for next-token prediction. The PINN constraint prevents this pruning by maintaining gradient pressure on M -correlated dimensions.

Implications for alignment. While this paper does not directly address AI safety, the mechanism we demonstrate—imposing structural constraints on a model’s internal trajectory during training—is relevant to the alignment research program. PINN constraints operate on the geometry of the learned representation itself, offering a potential path toward training-time structural guarantees on internal model properties.

6 Limitations

Scale. Our largest model is 25M parameters, far below frontier scale. While the cost-shrinks-with-scale trend suggests PINN constraints will remain viable at larger scale, we have not verified this beyond 25M.

Tokenization. All experiments use character-level or byte-level tokenization. Subword tokenizers (BPE, SentencePiece) are standard in production LLMs and may interact differently with the integration token definition.

Target signal. We test a single target signal family (sentence-ending punctuation count and its decay variant). We have not tested semantically richer targets.

Seeds. We report $n=3$ seeds per cell. While sufficient for variance estimation, $n \geq 5$ would strengthen statistical claims.

Probe methodology. Linear probes may understate the recoverability of nonlinearly encoded features. Our r^2 values should be interpreted as lower bounds on the information present in the residual stream.

7 Future Work

Future work will investigate the persistence properties of these constraints under continued training, their interaction with larger model scales, and extensions to safety-relevant trajectory specifications.

8 Conclusion

We demonstrate that physics-informed trajectory constraints can be imposed on a transformer language model’s residual stream during training at negligible cost to language modeling performance. Across two datasets, two model scales, four training conditions, and multiple seeds, we find that the approach is robust, generalizable across constraint types, and produces unexpected representational benefits—expanding effective dimensionality rather than compressing it, and preserving features that unconstrained training actively degrades. Our results establish PINN-style trajectory constraints as a viable mechanism for shaping transformer representations at training time, complementing post-hoc interpretability and steering approaches. The technique opens a path toward training-time structural guarantees on the internal properties of language models.

References

- Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.
- Ronen Eldan and Yuanzhi Li. Tinstories: How small can language models be and still speak coherent English? *arXiv preprint arXiv:2305.07759*, 2023.
- Edward S Hu et al. The belief state transformer. *arXiv preprint arXiv:2410.23506*, 2024.
- Andrej Karpathy. nanogpt. <https://github.com/karpathy/nanoGPT>, 2022.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2017.
- Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- Joseph Sill. Monotonic networks. In *Advances in Neural Information Processing Systems*, volume 10, 1997.
- Samarth Sinha et al. Next-latent prediction transformers learn compact world models. *arXiv preprint arXiv:2511.05963*, 2025.
- Adly Templeton et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Anthropic Research*, 2024.
- Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, et al. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023.