One-Step Generative Priors for LLM Activations

Anonymous ACL submission

Abstract

We adapt one-step average-velocity objectives to activation priors for LLM hidden states, allowing activation denoising to be performed with a single prior call rather than multi-step numerical integration. This preserves the benefits of GLP-style on-manifold steering while reducing the per-token denoising overhead by roughly an order of magnitude. Empirically, our method produces reliable activation reconstructions in the one-step regime and matches or improves multi-step GLP steering performance at substantially lower latency.

1 Introduction

Large language models encode rich information in their hidden states. This makes it possible to steer model behavior by adding a direction in activation space, e.g. $\boldsymbol{x}' = \boldsymbol{x} + \alpha \boldsymbol{s}$. Such interventions are simple and efficient, but the resulting activations may leave the natural activation manifold, which can reduce generation quality and make strong steering unstable.

Recent work addresses this by learning an activation prior over LLM hidden states (Luo et al., 2026). In particular, flow-matching priors can denoise a perturbed activation and move it back toward the distribution of real hidden states (Lipman et al., 2023). However, these priors usually require many denoising steps. This is expensive in LLM inference, since the procedure must be applied at every generated token.

In this work, we adapt MeanFlow-style average-velocity modeling to LLM activations (Geng et al., 2025a). Unlike prior flow-matching activation priors that learn an instantaneous velocity field and rely on numerical integration, our priors directly predict the interval-average velocity between two time points. This gives a single-step update for activation reconstruction and steering, without a separate distillation stage. Thus, our goal is not only to improve activation-space reconstruction, but to make GLP-style on-manifold steering practical by replacing many per-token denoising calls with a single average-velocity update.

We compare Euler and semi-group training conditions (Guo et al., 2026), with and without self-conditioning (Chen et al., 2022; Dieleman et al., 2022). Our experiments show that average-velocity priors produce reliable hidden states in the one-step regime, while ordinary flow matching requires many sampling steps. In on-manifold steering, our one-step method improves over vanilla steering and achieves performance comparable to multi-step GLP-style denoising, while being much faster.

Our contributions are: **(1)** we adapt established average-velocity objectives to GLP-style LLM activation priors, enabling direct one-step activation reconstruction without progressive distillation. **(2)** We compare Eulerian and semi-group average-velocity training conditions, with and without self-conditioning, in the LLM activation setting. **(3)** We show that one-step denoising improves on-manifold steering over vanilla steering and achieves performance comparable to multi-step GLP while substantially reducing per-token inference cost.

2 Background

2.1 Activation Priors for LLM Activations

Luo et al. (2026) learn an activation prior with flow matching (Lipman et al., 2023). For activation $\boldsymbol{x}$ and Gaussian noise $\boldsymbol{\varepsilon}$, they use the linear path

$$\boldsymbol{z}_t = (1-t)\boldsymbol{x} + t\boldsymbol{\varepsilon}, \boldsymbol{v}^\star(\boldsymbol{z}_t, t) \quad \triangleq \dot{\boldsymbol{z}}_t = \boldsymbol{\varepsilon} - \boldsymbol{x},$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$ and $t \sim \mathrm{Uniform}(0,1)$. The prior is trained to predict this velocity:

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\varepsilon},t} \left\| \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) - \boldsymbol{v}^\star(\boldsymbol{z}_t, t) \right\|_2^2 . \quad (1)$$

At inference time, samples are generated by integrating backward from $\boldsymbol{z}^{(0)} = \boldsymbol{\varepsilon}$. Since $\boldsymbol{v}_{\boldsymbol{\theta}}$ es-

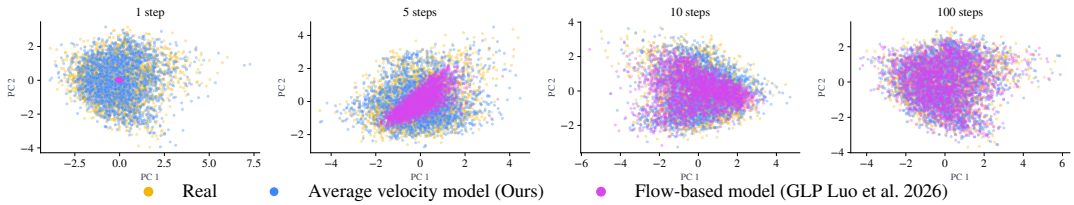

Figure 1: PCA of generated and original hidden states. Average velocity (Improved MeanFlow) produces reliable one-step samples, while Flow requires at least 10 steps to match real LLaMA-3.1-8B activations.

timates an instantaneous derivative, reliable sampling typically requires many steps, which is costly when denoising is applied at every generated token.

2.2 Steering with Meta-Models

Activation steering exploits linear structure in LLM hidden states (Elhage et al., 2022; Park et al., 2024a) by adding a direction $\boldsymbol{s}$ to an activation, $\boldsymbol{x}' = \boldsymbol{x} + \alpha \boldsymbol{s}$, where α controls steering strength. Such directions are often obtained with SAEs (Elhage et al., 2022) or contrastive methods such as persona vectors (Chen et al., 2025). A trained generative prior can further refine a steered activation by noising $\boldsymbol{x}'$ to $\boldsymbol{z}'_{t_n} = (1 - t_n)\boldsymbol{x}' + t_n \boldsymbol{\varepsilon}$ and denoising with $\boldsymbol{v}_{\boldsymbol{\theta}}$ to obtain final $\widetilde{\boldsymbol{x}}$.

3 Improving Flow Matching for LLM Activations

3.1 Self-Conditioning

Self-conditioning was introduced for image diffusion by Chen et al. (2022) and is also widely used in continuous diffusion models for categorical data (Dieleman et al., 2022). In typical sampling scheme the model discards much of its intermediate information after each step, including its estimate of the clean activation $\hat{\boldsymbol{x}}_0$. Self-conditioning addresses this by feeding this estimate back into the model during both training and sampling.

During training, we first estimate the clean activation and then condition the velocity prediction on it:

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{z}_t - t\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t, \varnothing),$$
$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \mathbb{E}\,\|\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t, \mathrm{sg}(\hat{\boldsymbol{x}}_0)) - \boldsymbol{v}^{\star}\|_2^2\,,$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. We also drop the self-conditioning input with probability p_{sc}, so the model remains accurate without it. In all experiments, we set $p_{\mathrm{sc}} = 0.5$.

3.2 Average Velocity

As discussed in Section 2.1, instantaneous velocity fields require many Euler steps. MeanFlow (Geng et al., 2025a) instead learns the average noising velocity over an interval:

$$\boldsymbol{u}(\boldsymbol{z}_t, r, t) \triangleq \frac{1}{t-r}\int_r^t \boldsymbol{v}^{\star}(\boldsymbol{z}_\tau, \tau)\, d\tau. \qquad (2)$$

Here, t is the starting time and r is the target time. Since $\boldsymbol{u}$ points in the noising direction, one-step denoising subtracts it:

$$\boldsymbol{z}_r = \boldsymbol{z}_t - (t-r)\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, r, t).$$

A boundary condition is obtained at $r = t$, where the average velocity becomes the instantaneous velocity: $\boldsymbol{u}(\boldsymbol{z}_t, t, t) = \boldsymbol{v}^{\star}(\boldsymbol{z}_t, t)$.

All methods below include the corresponding boundary loss,

$$\mathcal{L}_{\mathrm{boundary}}(\boldsymbol{\theta}) = \|\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t, t) - \boldsymbol{v}^{\star}(\boldsymbol{z}_t, t)\|_2^2\,. \quad (3)$$

The boundary loss only trains the diagonal case $r = t$, so we add off-diagonal consistency conditions for intervals with $r \neq t$.

Naively supervising $\boldsymbol{u}(\boldsymbol{z}_t, r, t)$ would require integrating the unknown velocity field for every interval. We avoid this by training with consistency conditions that define targets from nearby model evaluations.

Semi-group condition (Guo et al., 2026). For $r < s < t$, additivity of the integral in Equation 2 gives

$$(t-r)\boldsymbol{u}(\boldsymbol{z}_t, r, t) = (s-r)\boldsymbol{u}(\boldsymbol{z}_s, r, s) + (t-s)\boldsymbol{u}(\boldsymbol{z}_t, s, t).$$

Here the intermediate state is reached by the same backward update,

$$\boldsymbol{z}_s = \boldsymbol{z}_t - (t-s)\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, s, t),$$

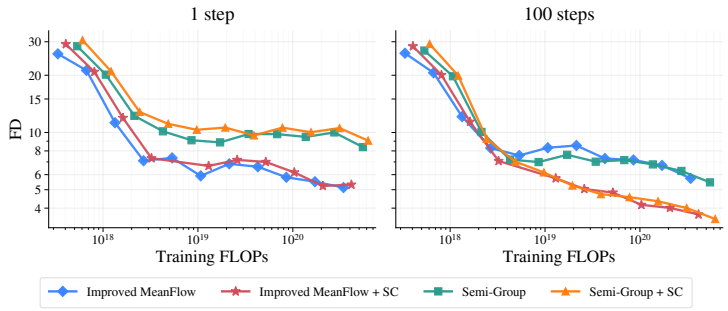

Figure 2: Training FLOPs and Frechet distance (FD) across methods. The Euler-based method performs best in the one-step setting. As the number of sampling steps increases, methods with self-conditioning produce better hidden states. See Section 4.1 for details.

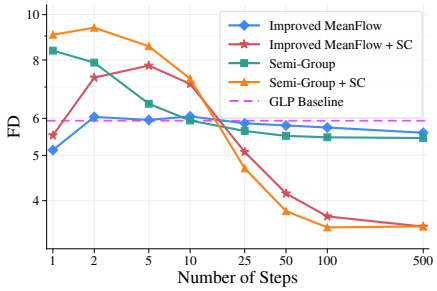

Figure 3: Frechet distance (FD) versus sampling steps. Euler methods perform best at one step, while self-conditioning methods dominate at larger budgets.

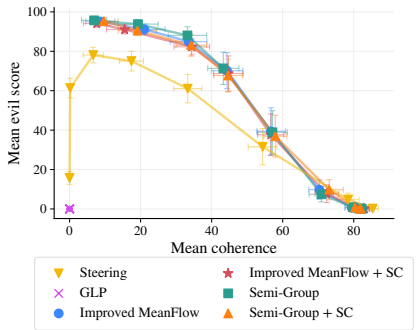

Figure 4: One-step on-manifold steering performance. The cross at $(0, 0)$ marks the original GLP model (Luo et al., 2026), which fails to reconstruct reliable hidden states in the one-step setting. Average-velocity methods perform similarly and outperform the vanilla steering baseline. See Section 4.3 for details.

or by the corresponding exact point on the linear path during supervised training. With $\lambda = (t - s)/(t - r)$, the semi-group target is

$$\boldsymbol{u}_{\text{target}} \triangleq (1 - \lambda)\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_s, r, s) + \lambda \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, s, t).$$

The training loss is

$$\mathcal{L}_{\text{semi}}(\boldsymbol{\theta}) = \|\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, r, t) - \text{sg}(\boldsymbol{u}_{\text{target}})\|_2^2 + \mathcal{L}_{\text{boundary}}(\boldsymbol{\theta}).$$

Eulerian condition (Geng et al., 2025a). Differentiating Equation 2 with respect to t gives

$$\boldsymbol{u}(\boldsymbol{z}_t, r, t) + (t - r)\frac{d}{dt}\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \boldsymbol{v}^\star(\boldsymbol{z}_t, t). \quad (4)$$

Writing this total derivative as the JVP shorthand $\texttt{JVP}_{\text{sg}}(\boldsymbol{u}, \boldsymbol{v})$ (see Appendix A) and taking $\boldsymbol{v} = \boldsymbol{v}^\star$, this yields the target

$$\boldsymbol{u}_{\text{target}} = \boldsymbol{v}^\star(\boldsymbol{z}_t, t) - (t - r)\texttt{JVP}_{\text{sg}}(\boldsymbol{u}, \boldsymbol{v}),$$

with loss
$\mathcal{L}_{\text{MF}}(\boldsymbol{\theta}) = \|\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, r, t) - \text{sg}(\boldsymbol{u}_{\text{target}})\|_2^2$.

A related variant is **Improved Mean Flow** (Geng et al., 2025b), which predicts the implied instantaneous velocity

$$\boldsymbol{V}_{\boldsymbol{\theta}} = \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, r, t) + (t - r)\texttt{JVP}_{\text{sg}}(\boldsymbol{u}, \boldsymbol{v}),$$

where the JVP uses $\boldsymbol{v} = \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t, t)$. The model is trained with the standard flow-matching loss $\mathcal{L}_{\text{IMF}} = \|\boldsymbol{V}_{\boldsymbol{\theta}} - \boldsymbol{v}^\star\|_2^2$.

We find that this variant trains more stably and outperforms the vanilla Eulerian condition, see Appendix A.2.

We also tested a Lagrangian condition (Boffi et al., 2025), whose one-step performance was similar to the base model; see Appendix A.2 and D.3. The Eulerian and Lagrangian conditions are infinitesimal endpoint limits of the semi-group condition, while shortcut and consistency models impose related interval-consistency constraints (Frans et al., 2025; Song et al., 2023). We therefore use the semi-group objective as our representative interval-consistency baseline, avoiding additional architectural confounders.

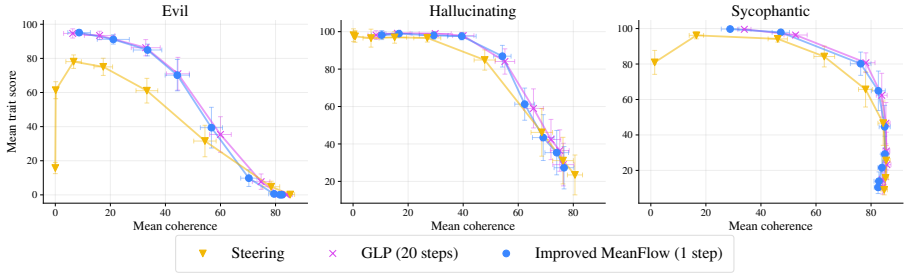

Figure 5: Comparison between the proposed one-step on-manifold steering method and GLP with 20 denoising steps, as used in Luo et al. (2026). We also report 95% CI to show robustness and significance of the results.

4 Experiments

We evaluate whether average-velocity activation priors can remove the multi-step inference bottleneck of GLP-style denoising. Specifically, we ask whether a single average-velocity update can match many-step flow matching in activation quality, preserve the benefits of on-manifold steering, and reduce per-token generation latency. Figure 1 gives a qualitative PCA view of the generated hidden states.

4.1 Training Efficiency

Following Luo et al. (2026), our main experiments use LLaMA-3.1-8B activations. See Appendix B for data preparation details. We save checkpoints on a logarithmic grid and report activation-space Frechet distance (FD) against estimated training FLOPs. Figure 2 shows that Euler-based methods perform best in the one-step setting, while self-conditioning helps at larger sampling budgets. Although the semi-group condition is theoretically more costly, in practice we observe only a negligible wall-time slowdown of about 20%.

4.2 Generation Quality

Using the final checkpoints for LLaMA-8B, we measure FD across different number of steps and compare against the GLP baseline. All runs use the public 50K-activation reference set from Luo et al. (2026). Analogous reconstruction results for LLaMA-1B are provided in Appendix D.1.

Results are presented in Figure 3. Self-conditioning improves larger-budget generation quality, while non-SC variants remain closer to the GLP baseline as the number of inference steps increases. In the one-step regime, Euler-based methods outperform the 1000-step GLP baseline in FD, suggesting that activation reconstruction can be reduced to one denoising step without sacrificing activation-space quality.

4.3 On-Manifold Steering

We follow the persona-vector setup of Luo et al. (2026); Chen et al. (2025). We use the *evil* vector to compare average-velocity objectives, and *evil*, *hallucinating*, and *sycophantic* to compare one-step Euler against 20-step GLP. Each trait uses 20 questions, 320 completions, and 95% confidence intervals; generation details are in Appendix E.

Figure 4 shows that all average-velocity objectives perform similarly in steering and outperform vanilla steering, despite differences in FD. The original GLP model (Luo et al., 2026) fails in the one-step setting, producing near-zero hidden states and effectively random outputs.

Method	Tok/s ↑	Slowdown ↓
Steering	1118	1.0×
1 Step (Ours)	894	1.25×
20 Steps (GLP)	83	13.47×

Table 1: Generation speed and slowdown.

Figure 5 shows that one-step Euler matches or exceeds the original 20-step GLP baseline while being several times faster. For length-32 prompts and up to 128 new tokens, Table 1 shows that one-step denoising is about 10× faster than 20-step GLP, with only a modest slowdown over vanilla steering.

5 Conclusion

We show that average-velocity activation priors remove the multi-step denoising bottleneck in GLP-style on-manifold steering. By predicting interval-average velocities, our method reconstructs LLM activations with a single update, substantially reducing latency while preserving the benefits of activation-prior denoising. These results suggest that one-step generative priors are a practical route to activation-space control during LLM inference.

6 Limitations

Although we evaluate our method on both LLaMA-3.2-1B (see Appendix D.1) and LLaMA-3.1-8B activations, the experiments remain limited to a single model family. These results therefore do not establish that one-step activation priors transfer unchanged to other architectures, training distributions, hidden-state geometries, or substantially larger models. Our main experiments also focus on a fixed activation layer in the residual stream, so the behavior of average-velocity priors at other layers, attention or MLP subspaces, and different intervention sites remains open.

Our strongest cross-scale evidence is in activation reconstruction, where the LLaMA-1B results show that MeanFlow-style priors substantially improve over one-step GLP in Frechet distance and exhibit behavior consistent with the 8B results. However, the full on-manifold steering evaluation is still concentrated on the 8B setting. The steering experiments also cover three persona vectors and a fixed prompt set, which supports controlled comparison with prior GLP-style experiments but leaves broader task domains, steering directions, and prompt distributions for future work.

Finally, our method reduces the denoising bottleneck but does not remove it entirely. One-step denoising still requires an additional activation-prior call at each generated token, which may be costly in very latency-sensitive or memory-constrained deployments. Self-conditioning also remains only partially explored: while it improves larger-budget generation quality, we do not yet fully exploit it for one-step steering.

7 Ethical Considerations

This work studies methods for modifying LLM behavior through activation steering. Such methods can be used constructively, for example to improve controllability, reduce unwanted behaviors, or study internal representations. However, the same techniques could also be used to amplify harmful, deceptive, sycophantic, or otherwise undesirable behaviors. Our experiments include persona vectors such as *evil*, *hallucinating*, and *sycophantic* only as controlled stress tests for evaluating whether denoising methods preserve or modify steering effects. These experiments should not be interpreted as recommending deployment of such behaviors. Any practical use of activation steering or activation denoising should include safety evaluations, monitoring of downstream generations, and safeguards appropriate to the target application.

References

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. 2025. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Trans. Mach. Learn. Res.*, 2025.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. 2025. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv: 2507.21509*.

Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. 2022. Analog bits: Generating discrete data using diffusion models with self-conditioning. *International Conference on Learning Representations*.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. 2022. Continuous diffusion for categorical data. *arXiv preprint arXiv: 2211.15089*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/toy_model/index.html.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. 2025. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. 2025a. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. 2025b. Improved mean flows: On the challenges of fast-forward generative models. *arXiv preprint arXiv: 2512.02012*.

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, and Yuxuan Wang. 2026. Splitmeanflow: Interval splitting consistency in few-step generative modeling.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2023. Flow matching for generative modeling. In *International Conference on Learning Representations*. OpenReview.net.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Grace Luo, Jiahai Feng, Trevor Darrell, Alec Radford, and Jacob Steinhardt. 2026. Learning a generative meta-model of llm activations. *arXiv preprint arXiv: 2602.06964*.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024a. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39643–39666. PMLR.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024b. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *arXiv preprint arXiv:2303.01469*.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Llama team. 2024. The llama 3 herd of models. *arXiv preprint arXiv: 2407.21783*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *arXiv preprint arXiv: 2308.10248*.

A Deriving Targets for MeanFlow.

For simplicity in the main paper, we fix the definition of the Jacobian-vector product as

$$\texttt{JVP}_{\text{sg}}(\boldsymbol{u}, \boldsymbol{v}) \triangleq \partial_{\boldsymbol{z}} \boldsymbol{u}(\boldsymbol{z}_t, r, t)\boldsymbol{v} + \partial_t \boldsymbol{u}(\boldsymbol{z}_t, r, t),$$

for convenience in the Eulerian condition. However, in the following section, we use $J_{\boldsymbol{u}}(\boldsymbol{z}_t, r, t)[\cdot]$ to denote the Jacobian-vector product of $\boldsymbol{u}$ with respect to all of its inputs.

A.1 Eulerian

Starting from the definition,

$$(t-r)\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \int_r^t \boldsymbol{v}^\star(\boldsymbol{z}_\tau, \tau)\, d\tau.$$

Differentiating with respect to t:

$$\boldsymbol{u}(\boldsymbol{z}_t, r, t) + (t-r)\frac{d}{dt}\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \boldsymbol{v}^\star(\boldsymbol{z}_t, t)$$

Since the inputs of the mean velocity field depend on t, we need to compute the total derivative of the composite function $\boldsymbol{u}(\boldsymbol{z}_t, r, t)$. By the chain rule,

$$\frac{d}{dt}\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{z}_t}\frac{\partial \boldsymbol{z}_t}{\partial t} + \frac{\partial \boldsymbol{u}}{\partial r}\frac{\partial r}{\partial t} + \frac{\partial \boldsymbol{u}}{\partial t}\frac{\partial t}{\partial t}.$$

Here r is fixed, so $\partial r/\partial t = 0$, while $\partial t/\partial t = 1$. Moreover, along the flow trajectory, $\partial \boldsymbol{z}_t/\partial t = \boldsymbol{v}^\star(\boldsymbol{z}_t, t)$. Therefore,

$$\frac{d}{dt}\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{z}_t}\boldsymbol{v}^\star(\boldsymbol{z}_t, t) + \frac{\partial \boldsymbol{u}}{\partial t}.$$

This total derivative can be written compactly as a Jacobian-vector product. Let $\boldsymbol{y} = (\boldsymbol{z}_t, r, t)$ denote the full input to $\boldsymbol{u}$. When differentiating with respect to t, this input changes in the direction

$$\frac{d\boldsymbol{y}}{dt} = \left(\frac{d\boldsymbol{z}_t}{dt}, \frac{dr}{dt}, \frac{dt}{dt}\right) = (\boldsymbol{v}^\star(\boldsymbol{z}_t, t), 0, 1)\,.$$

Hence,

$$\frac{d}{dt}\boldsymbol{u}(\boldsymbol{z}_t, r, t) = J_{\boldsymbol{u}}(\boldsymbol{z}_t, r, t)\left[\boldsymbol{v}^\star(\boldsymbol{z}_t, t), 0, 1\right],$$

where $J_{\boldsymbol{u}}(\boldsymbol{z}_t, r, t)[\cdot]$ denotes the Jacobian-vector product of $\boldsymbol{u}$ with respect to all of its inputs. The tangent used in the JVP is therefore $(\boldsymbol{v}^\star, 0, 1)$: the first component moves $\boldsymbol{z}_t$ along the flow trajectory, the second keeps r fixed, and the third increases t at unit speed.

Plugging this into Equation 4, we obtain

$$\boldsymbol{u}(\boldsymbol{z}_t, r, t) + (t-r)J_{\boldsymbol{u}}\left[\boldsymbol{v}^\star(\boldsymbol{z}_t, t), 0, 1\right] = \boldsymbol{v}^\star(\boldsymbol{z}_t, t).$$

Therefore

$$\boldsymbol{u}_{\text{target}} = \boldsymbol{v}^\star(\boldsymbol{z}_t, t) - (t-r)J_{\boldsymbol{u}}\left[\boldsymbol{v}^\star(\boldsymbol{z}_t, t), 0, 1\right].$$

A.2 Lagrangian Condition

Recall average velocity definition

$$(t-r)\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \int_r^t \boldsymbol{v}^\star(\boldsymbol{z}_\tau, \tau)\, d\tau.$$

differentiating with respect to endpoint r:

$$\boldsymbol{u}(\boldsymbol{z}_t, r, t) = \boldsymbol{v}^\star(\boldsymbol{z}_r, r) + (t-r)\frac{d}{dr}\boldsymbol{u}(\boldsymbol{z}_t, r, t).$$

Note that

$$\begin{aligned}\frac{d}{dr}\boldsymbol{u}(\boldsymbol{z}_t, r, t) &= \partial_r \boldsymbol{u}(\boldsymbol{z}_t, r, t) \\ &= J_{\boldsymbol{u}}(\boldsymbol{z}_t, r, t)\left[\mathbf{0}, 1, 0\right].\end{aligned}$$

We can now define

$$\begin{aligned}\boldsymbol{u}_{\text{target}} = \boldsymbol{v}^\star(\boldsymbol{z}_r, r) \\ + (t-r)J_{\boldsymbol{u}}(\boldsymbol{z}_t, r, t)\left[\mathbf{0}, 1, 0\right].\end{aligned}$$

training loss becomes

$$\mathcal{L}_{\text{lagrange}} = \|\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{u}_{\text{target}}\|_2^2.$$

B Justification on Data Repetition

Method	Baseline FD	Our Pipeline FD
GLP 3 Layers	0.68	0.71
GLP 6 Layers	0.61	0.63
GLP 9 Layers	0.55	0.59
GLP 12 Layers	0.53	0.54

Table 2: Effect of data repetition on Frechet distance.

During our preliminary experiments, we trained models using the two-stage pipeline described in Section 4.1 on the Llama-1B model with different numbers of layers and compared them against publicly available checkpoints from Luo et al. (2026). We report Frechet distance in Table 2. Please note that the original data processing pipeline was not available at the time of the experiments. However, for Frechet distance validation, we used a publicly available reference set of 50K activations. Small details, such as BOS token removal or sequence packing, were also unavailable, so the mismatch could partially be explained by minor inconsistencies between the training and validation data. Nevertheless, our reproduction and data repetition results differ by no more than 0.04 in Frechet distance.

C Related Work

Activation steering. Activation steering modifies hidden states at inference time to control model behavior without updating model weights. Early work showed that latent steering vectors can be extracted directly from pretrained language models and added to hidden states to control generation (Subramani et al., 2022). Later activation-engineering methods made this intervention simpler and more practical: Activation Addition computes steering vectors from prompt pairs (Turner et al., 2023), Inference-Time Intervention shifts activations along truthfulness-related directions (Li et al., 2023), and Contrastive Activation Addition computes directions from positive and negative behavioral examples (Rimsky et al., 2024). Our work uses the same broad intervention paradigm, but focuses on reconstructing steered activations so that they remain close to the activation manifold.

Linear representations and feature directions. The linear representation hypothesis suggests that high-level concepts can often be represented as directions in activation space (Park et al., 2024b,a). This view is also related to work on sparse autoencoders, which learn interpretable feature directions from LLM activations (Cunningham et al., 2023; Elhage et al., 2022). Such methods motivate simple additive interventions, but they do not by themselves guarantee that the modified activation remains in-distribution. Our method is complementary: it can be applied after a steering direction is chosen, regardless of how that direction was obtained.

Generative models of LLM activations. Luo et al. (2026) proposed learning a generative latent prior over LLM activations using flow matching (Lipman et al., 2023). This allows perturbed or steered activations to be denoised before being passed back into the language model. However, standard flow matching learns an instantaneous velocity field and usually requires many numerical integration steps. This is costly for activation-space applications, where denoising may be applied at every generated token.

Fast sampling in generative models. Reducing the number of sampling steps is a central problem in diffusion and flow-based generative modeling. Consistency models learn mappings that support one-step or few-step generation (Song et al., 2023), rectified flows aim to learn straighter transport paths that can be sampled with coarse discretization (Liu et al., 2022), and progressive distillation reduces the number of diffusion sampling steps by repeatedly distilling a slower sampler (Salimans and Ho, 2022). MeanFlow-style models instead learn average velocities over time intervals (Geng et al., 2025a), with semi-group variants enforcing consistency across sub-intervals (Guo et al., 2026). We adapt these ideas to LLM activation modeling and evaluate them in the one-step regime for on-manifold steering.

D Additional Results

D.1 LLaMA 1B Results

To verify that our findings transfer across model scale, we additionally train MeanFlow models on activations from the LLaMA-1B model. Specifically, we use activations from LLaMA 3.2 1B (team, 2024) and evaluate the same integration variants considered in the main experiments: Euler, improved Euler, and semi-group integration, with and without self-conditioning (SC). We report FD for both 1-step and 100-step generation and compare against the corresponding GLP baseline. Results presented in Table 3.

Method	1 step	100 steps
GLP baseline	18.26	0.70
MeanFlow	0.70	0.64
IMF	0.63	0.72
Semi-group	0.83	0.57
MeanFlow + SC	0.75	0.44
IMF + SC	**0.62**	0.44
Semi-group + SC	0.95	**0.39**

Table 3: FD results for the LLaMA-1B model.

The results show that MeanFlow substantially improves over the GLP baseline in the 1-step setting on LLaMA-1B activations. While the GLP baseline obtains an FD of ~ 18 with one sampling step, all MeanFlow variants achieve FD below 1.0, with improved Euler + SC performing best at 0.62. At 100 steps, self-conditioning consistently improves performance across all integration variants. In particular, semi-group + SC achieves the best overall FD of 0.39, outperforming both the GLP baseline and the non-self-conditioned semi-group variant. These results suggest that the gains from MeanFlow are not limited to the original model

scale and transfer to LLaMA-1B activations.

D.2 Mean Flow and Improved Mean Flow

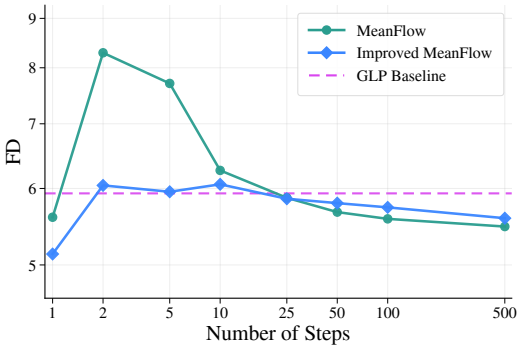

Figure 6: MeanFlow and Improved MeanFlow Frechet distance (FD) comparison.

As a representative of the Eulerian-based methods we chose Improved MeanFlow (IMF) (Geng et al., 2025b), however, in our preliminary experiments we used ordinary MeanFlow (Geng et al., 2025a). While IMF is known for stable gradients we do not observe instabilities in both methods after applying normalization on the losses. However, metrics for the IMF in terms of FD in one-step generation setup has improved. See Figure 6. Therefore we decide to include only Improved Mean Flow in the main text of the experiments. These methods are closely related, since both arise from the same Eulerian consistency condition.

D.3 Lagrangian Consistency

As mentioned in Section 3.2, we found that the Lagrangian consistency condition leads to performance similar to the base flow model. We use the setup from Section 4.1 and train two baseline models, with and without self-conditioning, as well as two models with the Lagrangian consistency loss. Results are presented in Table 4.

Model	1 step	100 steps
GLP	97.84	6.66
GLP + SC	97.21	4.71
Lagrangian	98.08	6.33
Lagrangian + SC	97.41	4.45

Table 4: Baseline Frechet distance (FD) values for GLP and Lagrangian models.

Our first observation is that self-conditioning is beneficial for all training types, especially in the many-step setup. While Lagrangian models have slightly better results at 100 steps, the results in the one-step setup are very similar.

E Hyperparameters and Additional Information for Reproducibility

Hyperparameter	Value
Training activation layer	16 (index 15)
Dataset	FineWeb
Hidden dimension	4096
Denoiser width	8192
MLP width	16384
Denoiser depth	6 layers
Activation function	SiLU
Training epochs	10
Batch size	4096
Optimizer	AdamW
Learning rate	5×10^{-5}
Warmup ratio	0.01
Gradient clipping	1.0
Precision	bfloat16
Time sampling	Uniform $[0, 1]$
Diagonal / boundary batches	75%
Off-diagonal batches	25%
Loss normalization	p_{norm}
Loss stabilizer ϵ	0.01
Eulerian p_{norm}	1.0
Lagrangian p_{norm}	0.0
Self-conditioning probability	0.5

Table 5: Hyperparameters used for the MeanFlow reproduction experiments.

Data preparation. We first obtain LLaMA 8B activations from layer 16 of the model. We filter out the first tokens of each sequence, as they tend to have higher norms. We also do not apply sequence packing, which we found leads to more stable training. Tokens are obtained from the FineWeb dataset (Penedo et al., 2024). We found that shuffling tokens is necessary to obtain smooth loss curves during flow matching training. Following Luo et al. (2026), we fix the maximum sequence length to 2048.

Training pipeline. Relative to the public GLP implementation of Luo et al. (2026), we retain the activation-space MLP denoiser, activation normalization scheme, and cached-activation training setup, but replace the original flow-matching train-

Hyperparameter	Value
Persona t_{start}	0.5
Persona MeanFlow steps	1
Samples per question	16
Maximum generated tokens	128
Judge model	`gpt-4o`

Table 6: Hyperparameters for persona vector experiment.

ing objective with a MeanFlow objective. The modified model conditions the denoiser on both the denoising time and the interval length, enabling direct one-step transport in activation space. We additionally add optional self-conditioning, where a bootstrapped clean-activation estimate is provided as an auxiliary input during training and inference. For the Eulerian and Lagrangian MeanFlow setups, we use the loss-normalization scheme from the original MeanFlow paper (Geng et al., 2025a): the per-sample regression loss is divided by a detached power-normalized loss scale, controlled by p_{norm}, with numerical stabilizer $\epsilon = 0.01$. Training pairs are sampled uniformly over time, with 75% of batches placed on the diagonal condition $r = t$, so the model sees a mixture of boundary velocity supervision and off-diagonal interval transport examples.

We provide hyperparameters needed to reproduce our training in Table 5.

Persona vectors experiment. In this experiment, we copy all default parameters from the original script. To ensure reproducibility, we provide hyperparameters in Table 6. We use steering coefficients $[0.5, 1.0, 1.5, 2.0, 2.5, 3.5, 4.0, 5.0]$.