Predictive Feature Caching for Training-free Acceleration of Molecular Geometry Generation

Johanna Sommer[1,2] **John Rachwan**[1] **Nils Fleischmann**[1]

Stephan Günnemann[1,2] **Bertrand Charpentier**[1]

[1] Pruna AI
[2] Technical University of Munich, Germany
{firstname.lastname}@pruna.ai

Abstract

Flow matching models generate high-fidelity molecular geometries but incur significant computational costs during inference, requiring hundreds of network evaluations. This inference overhead becomes the primary bottleneck when such models are employed in practice to sample large numbers of molecular candidates. This work discusses a training-free caching strategy that accelerates molecular geometry generation by predicting intermediate hidden states across solver steps. The proposed method operates directly on the SE(3)-equivariant backbone, is compatible with pretrained models, and is orthogonal to existing training-based accelerations and system-level optimizations. Experiments on the GEOM-Drugs dataset demonstrate that caching achieves a twofold reduction in wall-clock inference time at matched sample quality and a speedup of up to $3\times$ compared to the base model with minimal sample quality degradation. Because these gains compound with other optimizations, applying caching alongside other general, lossless optimizations yield as much as a $7\times$ speedup.

1 Introduction

Deep learning, particularly deep generative modeling, is rapidly transforming molecular design by enabling the de-novo creation of molecular geometries [Wang et al., 2025, Alakhdar et al., 2024, Tang et al., 2024]. Among generative methods, flow matching models have emerged as the state-of-the-art for generating high-quality molecular geometries. Their iterative denoising nature allows for flexible modeling of complex geometric distributions. While earlier approaches focused on SMILES strings or molecular graphs, the direct generation of molecular geometries has gained traction due to its fidelity in capturing geometric and physicochemical constraints critical for real-world efficacy.

Traditional drug discovery pipelines rely on combinatorial screening of known compounds. In contrast, generative models aim to sample directly from the underlying chemical distribution. Although this approach offers a more controlled way to navigate the chemical space than, for example, virtual or high-throughput screening [Johansson et al., 2024], such models in practice will still be used to sample 500,000 or even over 1 million compounds [Shen et al., 2024, Koziarski et al., 2024]. As a consequence, the molecular generator's inference time is the dominant bottleneck. This holds especially for diffusion or flow matching models, whose sampling can require hundreds of neural-network evaluations for a single molecule, resulting in prohibitively slow sampling at scale.

Accepted at the AI for Science workshop (NeurIPS 2025).

All practical acceleration methods that have been introduced specifically for molecule generation *require additional training*, incurring data, compute, and time overhead. Trajectory re-parameterization trains diffusion models to straighten their stochastic paths, thereby reducing the number of steps required to reach data-like samples [Ni et al., 2025]. Progressive distillation trains a student to replace two teacher denoising steps with one, iteratively halving the sampling budget [Lacombe and Vaidya, 2024]. Latent methods train an autoencoder and then a generator in the compressed space, allowing denoising to run in a lower-dimensional latent space and cutting per-step computation [Xu et al., 2023].

While architectural refinements and diffusion process adjustments have led to significant gains in the speed and efficiency of molecular generative models, we pursue a *complementary* and *training-free* direction. Inspired by recent advances in image generation, we investigate an acceleration scheme for molecular geometry generation models that reuses and forecasts intermediate hidden states during sampling. By leveraging previously computed features across time steps, our method reduces redundant computation, achieving up to a $3\times$ speedup in generation with minimal loss in sample quality.

Our contributions are as follows:

- We investigate predictive feature caching as the first training-free scheme that accelerates molecular geometry generation models at little to no quality loss. The method is drop-in for pretrained models and reduces inference cost while preserving generation quality.
- We transfer and analyze multiple feature-caching strategies to molecular geometry generation, operating directly on molecular geometries and SE(3)-equivariant representations.
- We show that our approach is complementary to general post-training optimizations, highlighting that they can be combined for significant inference time speedup.

2 Related Work

Diffusion caching. Caching was first introduced for image diffusion models, where prior work [Wimbauer et al., 2023, Ma et al., 2023, Li et al., 2024] observed temporal redundancy in the U-Net's high-level features, which can be exploited by caching and reusing them across successive denoising steps. Extending this to diffusion transformers (DiTs), Selvaraju et al. [2024] report analogous temporal similarity in attention and MLP activations and propose reusing them over multiple steps. Chen et al. [2024] leverages role asymmetries across blocks, caching rear blocks early and front blocks late via a DiT-specific Δ-cache. Rather than reusing features from the most recent step, TaylorSeer [Liu et al., 2025a] predicts future features via a Taylor-series expansion, while AB-Cache [Yu et al., 2025] employs an Adams–Bashforth scheme to compute a predictions based on previously computed features. Classical caching recomputes features every other step; in contrast, TeaCache adaptively decides when to refresh the cache based on the inputs of the DiT [Liu et al., 2025b]. In image generation, Region Adaptive Sampling allocates computation spatially, updating only regions in focus while reusing cached features elsewhere [Liu et al., 2025c]. For video generation models, caching must respect temporal coherence and inter-frame redundancy, hence various caching strategies specifically tailored to video generation have been proposed [Sun et al., 2025, Lv et al., 2025, Yuan et al., 2024, Liu et al., 2025b]. Previous work has primarily focused on various caching approaches for the image and video domains; none of which have been transferred to deep generative models in the molecular domain.

De-novo molecular geometry generation. Work on de-novo molecule generation today focuses on directly generating molecules as their 3D coordinates, representing structures in continuous Euclidean space, and parameterizing them as Cartesian or internal coordinates alongside atom types. Within this line of work, variational autoencoders learn a latent space over geometries and decode molecules with equivariant architectures that enforce basic geometric constraints [Ragoza et al., 2020]. Building on that, autoregressive models place atoms or fragments sequentially in a 3D space, conditioned on the growing partial structure [Gebauer et al., 2020, Luo and Ji, 2022]. In parallel, normalizing flow-based methods define invertible transformations over coordinates to provide exact likelihoods under $E(n)$ / $SE(3)$ equivariance [Satorras et al., 2022].

Most recently, diffusion-based approaches have become prominent: some follow classical score-based diffusion [Hoogeboom et al., 2022, Huang et al., 2022, 2023, Vignac et al., 2023, Qiang et al., 2023, Morehead and Cheng, 2024, Wu et al., 2022, Xu et al., 2023, Reidenbach et al., 2025, Hong et al., 2025, Feng et al., 2025, Ni et al., 2025, Irwin et al., 2025] while others adopt the closely related flow-matching formulation to learn continuous probability flows [Song et al., 2023, Dunn and Koes, 2024, Joshi et al., 2025]. Although such models generate molecules unconditionally, they can be adapted for downstream tasks, such as property optimization or shape-constrained generation, via, for example, diffusion guidance at inference time [Ayadi et al., 2025, Ketata et al., 2024].

3 Flow Matching for Molecular Geometry Generation

Let p_{data} denote a target data distribution on a state space $\mathcal{X}$ and let p_{noise} be a simple base distribution on $\mathcal{X}$. Flow matching transports p_{noise} at $t = 0$ to p_{data} at $t = 1$ by learning a time-dependent vector field $\{v_\theta(\cdot, t)\}_{t\in[0,1]}$ [Lipman et al., 2023]. v_θ induces a flow $\Phi_{s\to t}$ whose pushforward maps a probability path $(p_t)_{t\in[0,1]}$ from $p_0 = p_{\mathrm{noise}}$ to $p_1 = p_{\mathrm{data}}$ with the dynamics $\dot{x}_t = v_\theta(x_t, t)$.

Conditional flow matching (CFM) allows for training the vector field without simulating trajectories by specifying, for each data sample $x_1 \sim p_{\mathrm{data}}$, a conditional path distribution $\{p_{t|1}(\cdot \mid x_1)\}_{t\in[0,1]}$ and its conditional velocity field $u_t(x_t \mid x_1) \in T_{x_t}\mathcal{X}$, where $T_{x_t}\mathcal{X}$ denotes the tangent space of $\mathcal{X}$ at x_t [Tong et al., 2024]. For concreteness we only consider the condition x_1, but other conditioning choices are possible, for example an auxiliary variable defining a linear-interpolation bridge [Liu et al., 2022]. The training objective learns the velocity field by regressing

$$v_\theta(x_t, t) \;\approx\; u_t(x_t \mid x_1). \tag{1}$$

During sampling, we integrate this ODE with a discrete time grid $0 = t_0 < t_1 < \cdots < t_K = 1$. The first-order Euler scheme

$$x_{k+1} = x_k + \Delta t_k\, v_\theta(x_k, t_k), \qquad \Delta t_k := t_{k+1} - t_k, \tag{2}$$

instantiates the transformation from the base to the data distribution in K discrete steps.

Molecular geometry parameterization. Molecular geometries comprise multiple atoms, each with 3D coordinates and an atom type, and bonds between atoms labeled by discrete bond orders. We model a molecule as tuples

$$x = (c, a, b) \in \mathcal{X} := \underbrace{\mathbb{R}^{N\times 3}}_{\text{coord.}} \times \underbrace{\mathcal{A}^N}_{\text{atom types}} \times \underbrace{\mathcal{B}^{\mathcal{E}}}_{\text{bond orders}}, \tag{3}$$

and we learn the joint distribution $p(x)$. These variables are regressed jointly, yielding a single parameterization that induces a coupled vector field on coordinates, atom types, and bond orders. Given an atom count n, $x_0 \sim p_{\mathrm{noise}}(\cdot \mid n)$ is drawn and integrated from $t = 0$ to 1 as stated in Equation 2 to obtain joint samples $x = (c, a, b)$.

Equivariance. Molecular geometries are unchanged by global rotations and translations, so enforcing $E(3)$ equivariance prevents the model from learning spurious patterns. Let G act on the state space $\mathcal{X}$ via a representation $\rho_{\mathcal{X}} : G \to \mathrm{GL}(\mathcal{X})$. Here, $\rho_{\mathcal{X}}(g)\, x$ denotes the configuration obtained by applying g to x, for example by rotating or translating coordinates and permuting atoms. We denote the pushforward of this action on tangent vectors as $d\rho_{\mathcal{X}}$. A density p on $\mathcal{X}$ is G-invariant if $p(\rho_{\mathcal{X}}(g)\, x) = p(x)$ for all $g \in G$. A function $f : \mathcal{X} \to T\mathcal{X}$ is G-equivariant with respect to $(\rho_{\mathcal{X}}, d\rho_{\mathcal{X}})$ if

$$d\rho_{\mathcal{X}}(g)\, f(x) = f(\rho_{\mathcal{X}}(g)\, x) \quad \forall g \in G. \tag{4}$$

We enforce $E(3) \times S_N$ equivariance of the velocity field $v_\theta(\cdot, t)$. If the base density p_0 is G-invariant and $v_\theta(\cdot, t)$ is G-equivariant for all t, then the terminal density at $t = 1$ is G-invariant. For molecules we take $G = E(3) \times S_N$: the Euclidean group in 3D acting on coordinates and the symmetric group on N atoms acting by permutation on atoms. We enforce this by using isotropic coordinate noise and G-equivariant updates.

4 Predictive Feature Caching for the Molecular Domain

Evaluating the time-dependent vector field $v_\theta(x_t, t)$ dominates the inference cost of flow matching. The ODE solver has to query v_θ many times at closely spaced time steps, and each query runs the full backbone on inputs that change smoothly with t. As a result, intermediate activations at each network block evolve along a smooth feature trajectory over time.

In this work, we investigate *predictive feature caching* as a means to significantly reduce computational overhead at inference time by leveraging smooth feature trajectories during the generation of molecular geometries. Instead of recomputing similar features from scratch at every solver step, we store features at selected "checkpoint" times. We then reuse or predict features at nearby times to avoid full forward passes.

Recall from section 3 that we sample by integrating $\dot{x}_t = v_\theta(x_t, t)$ with $x_t = (c, a, b)$, where v_θ is implemented by a shared backbone. Let the backbone be a composition of functions $g_L \circ \cdots \circ g_1$. At solver time t, we denote the input to block l of L blocks in total by x_t^l and the block's output by $x_t^{l+1} := F^l(x_t^l)$. Because x_t evolves under an ODE with a smooth right-hand side and the network is continuous in (x, t), x_t^l will vary smoothly with t as shown in Figure 1. This smoothness provides a regularity that predictive caching exploits.

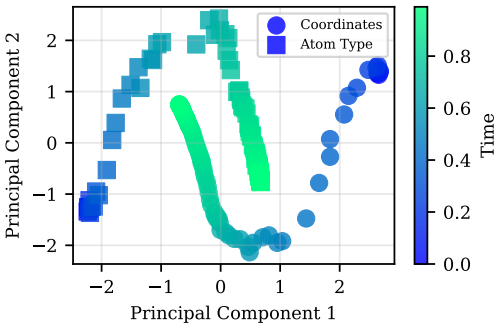

Figure 1: Projection onto the first two principal components of a single molecule's generation trajectory. Both coordinates and atom types evolve smoothly over solver steps.

Motivated by the strictly sequential information flow in transformer-based architectures for flow matching models and the high predictability of late-layer features, we follow Guan et al. [2025] and adopt a *last-block forecast*: at each timestep we apply the predictor only to block L, which avoids recomputation of the entire prefix $g_{1:L-1}$. For notational convenience, we continue by denoting the last block as $F := F^L$ and $x_t := x_t^L$.

TaylorSeer predictive caching. Naive caching simply reuses the last computed feature; it is cheap but accumulates staleness error because the features drift as t advances. Liu et al. [2025a] address this with TaylorSeer by turning caching into *predictive caching*: it leverages local Taylor expansions of the feature trajectory to forecast intermediate features. At predefined time steps spaced every D solver steps, we perform a standard forward pass and materialize the cache

$$C(x_t) = \{ F(x_t),\ \Delta F(x_t),\ \ldots,\ \Delta^m F(x_t) \}. \tag{5}$$

Using the Taylor–finite-difference correspondence, we can forecast features at an intermediate time $t + k$ by the m-th order predictor:

$$F_{\text{pred},m}(x_{t+k}) = F(x_t) + \sum_{i=1}^{m} \frac{\Delta^i F(x_t)}{i!\, D^i} (-k)^i, \tag{6}$$

which applies also to a first-order linear prediction and reduces to naive caching when $m = 0$.

Concretely, every D solver steps, we obtain $F(x_t)$ and populate the cache $C(x_t)$. For any timestep $t + k$ within the current window $\{t, \ldots, t+D\}$ we then forecast $F_{\text{pred},m}(x_{t+k})$. In our implementation, we guarantee, regardless of the caching interval D, that the last inference step is computed via $F(x_T)$.

Adams–Bashforth caching. Yu et al. [2025] similarly argues that the smooth feature trajectory can be exploited by applying a j-step Adams–Bashforth (AB) linear multi-step forecast. For flow matching, this yields the jth-order linear recursion

$$F_{\text{AB}(j)}(x_{t+k}) := \sum_{i=1}^{j} (-1)^{i+1} \binom{j}{i} F(x_{t+k+i}), \tag{7}$$

which uses the last j cached outputs to predict the current output. The implementation is similar to that of TaylorSeer caching; every D solver steps we populate the cache of backbone outputs, predict subsequent steps with $F_{\text{AB}(j)}(x_{t+k})$ and ensure that the last step is computed with $F(x_T)$.

Equivariance. Caching as we have established it is a time-scalar linear combinations of cached features and finite differences. These operations commute with the G action. Using the Euler update, equivariance yields the commutation relation

$$\rho_{\mathcal{X}}(g)\, x_{k+1} = \rho_{\mathcal{X}}(g)\, x_k + \Delta t_k\, d\rho_{\mathcal{X}}(g)\, v_\theta(x_k, t_k) = x'_{k+1} \quad \text{with} \quad x'_k = \rho_{\mathcal{X}}(g)\, x_k, \tag{8}$$

so each discrete step preserves the symmetry action. Consequently, if $v_\theta(\cdot, t)$ is G-equivariant at cached time steps and the base density p_{noise} is G-invariant (Sec. 3), the forecasted evaluations are G-equivariant throughout sampling, and the terminal density remains G-invariant. As a result, the discussed predictive feature caching preserves equivariance of the generation process.

5 Experiments

We evaluate caching on an equivariant flow-matching generator. Our primary goal is to characterize the quality–speed trade-off of two caching variants, Taylor forecasting and Adams–Bashforth (AB) multi-step, for molecular geometry generation. We do so by evaluating inference overhead alongside standard quality metrics. We identify operating points with no measurable quality loss and quantify the degradation–speed frontier when seeking higher acceleration. Additionally, we examine how caching composes with lossless, orthogonal optimizations.

Evaluation. We use two benchmark datasets, QM9 [Ramakrishnan et al., 2014] and GEOM Drugs [Axelrod and Gomez-Bombarelli, 2022], to assess a model's abilities as an unconditional molecular generator. Since QM9 contains only very small molecules, we employ GEOM Drugs a more meaningful benchmark for assessing model performance for real-world applications. The data splits correspond to those used in Vignac et al. [2023], Le et al. [2023].

As a base model, we utilize SemlaFlow [Irwin et al., 2025] and use pretrained weights provided by the authors for both the QM9 and GEOM Drugs dataset. Unless explicitly stated otherwise, we use the default hyperparameters of the model reported for each of the datasets. All metrics for SemlaFlow presented below are calculated by sampling from the distribution of molecule sizes in the test set, and then generating molecules with the sampled number of atoms by integrating the trained ODE with an Euler solver [Irwin et al., 2025]. SemlaFlow is not implemented with a fixed batch size, but rather sorts samples into buckets of similarly sized molecules to avoid significant padding for more efficient processing. Unless stated otherwise, we sample 10,000 molecules over 3 random seeds. All subsequent experiments are run on a single NVIDIA H100 PCIe GPU.

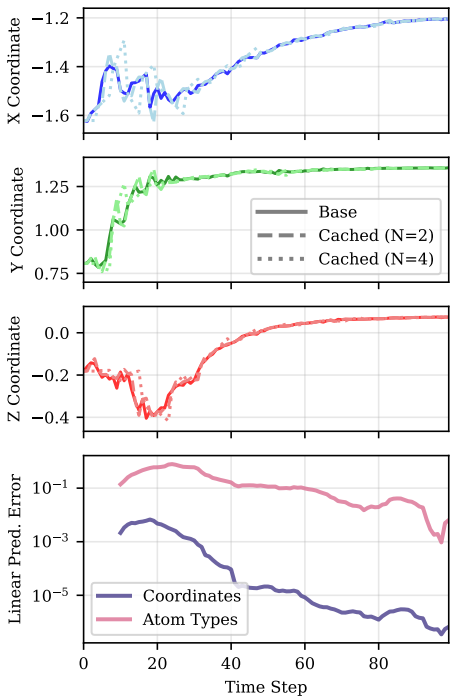

Figure 2: Trajectories of an atom's coordinates and evaluation of linear predictability error over sampling steps.

We evaluate sample quality using standard graph level metrics: novelty (fraction of generated molecules not seen in the training set), eniqueness (fraction of distinct molecules under canonical SMILES), atom stability (fraction of atoms with correct valence), and molecule stability (fraction of molecules for which all atoms are stable). As these topology-only metrics are saturated by current models and do not indicate conformation quality, we follow Irwin et al. [2025] and report (per-atom) energy and (per-atom) strain in kcal$*$mol^{-1}, where lower strain or energy indicates higher plausibility. To assess physical validity beyond topology we follow the evaluation from Buttenschoen et al. [2025] and report validity (Posebusters, RDKit, Connected = PRC) as the fraction of molecules that are connected, can be sanitized by RDKit [Landrum, 2013] and pass all Posebusters checks. Posebusters [Buttenschoen et al., 2024] requires passing checks on bond lengths/angles, planarity (aromatics and double bonds), steric clashes, and internal energy. Lastly, to measure inference time, we report throughput as the number of molecules that can be sampled per second.

Table 1: Evaluation of caching SemlaFlow trained on the GEOM Drugs dataset. We highlight in **bold** the best result per number of steps and underscore results that are better than or equal to the results of the base model at 100 steps.

Steps	Mode	Mol Stab. ↑	Valid (PRC) ↑	Energy ↓	Energy p.A ↓	Strain ↓	Strain p.A ↓	Opt. RMSD ↓	Throughput ↑
100	Base	0.98 ± 0.00	0.88 ± 0.01	108.8 ± 0.9	2.38 ± 0.01	69.6 ± 0.7	1.50 ± 0.01	0.86 ± 0.00	11.4 ± 0.1
51	Base	**0.98 ± 0.00**	0.86 ± 0.01	115.5 ± 0.8	2.51 ± 0.01	75.9 ± 0.8	1.63 ± 0.01	0.88 ± 0.00	21.9 ± 0.2
	Taylor $m = 1$	**0.98 ± 0.00**	0.85 ± 0.00	103.1 ± 1.2	2.28 ± 0.02	67.5 ± 1.0	1.48 ± 0.01	**0.87 ± 0.00**	21.8 ± 0.2
	Taylor $m = 2$	**0.98 ± 0.00**	0.86 ± 0.00	101.4 ± 0.7	2.25 ± 0.01	65.9 ± 0.2	1.46 ± 0.00	0.88 ± 0.00	22.1 ± 0.0
	AB $j = 2$	**0.98 ± 0.00**	0.85 ± 0.00	103.2 ± 1.0	2.28 ± 0.02	67.7 ± 0.9	1.49 ± 0.01	**0.87 ± 0.00**	21.8 ± 0.3
	AB $j = 3$	**0.98 ± 0.00**	**0.87 ± 0.00**	**96.5 ± 0.9**	**2.15 ± 0.01**	**62.8 ± 0.6**	**1.40 ± 0.01**	**0.87 ± 0.01**	22.1 ± 0.0
34	Base	**0.97 ± 0.00**	**0.85 ± 0.00**	120.3 ± 1.6	2.62 ± 0.03	82.0 ± 1.1	1.78 ± 0.02	0.90 ± 0.01	32.2 ± 0.6
	Taylor $m = 1$	**0.97 ± 0.00**	**0.85 ± 0.01**	103.8 ± 0.5	2.31 ± 0.01	70.0 ± 0.5	1.56 ± 0.01	**0.89 ± 0.00**	31.8 ± 0.5
	Taylor $m = 2$	**0.97 ± 0.00**	0.83 ± 0.00	100.9 ± 0.6	**2.25 ± 0.01**	68.2 ± 0.4	1.53 ± 0.00	**0.89 ± 0.00**	32.4 ± 0.1
	AB $j = 2$	**0.97 ± 0.00**	0.83 ± 0.00	105.5 ± 2.0	2.34 ± 0.04	70.0 ± 1.5	1.55 ± 0.03	0.90 ± 0.01	31.9 ± 0.4
	AB $j = 3$	**0.97 ± 0.00**	**0.85 ± 0.00**	**100.5 ± 1.0**	**2.25 ± 0.02**	**67.7 ± 0.7**	**1.51 ± 0.02**	0.90 ± 0.01	32.1 ± 0.3
26	Base	**0.97 ± 0.00**	**0.82 ± 0.00**	123.5 ± 0.4	2.69 ± 0.01	85.5 ± 0.5	1.85 ± 0.01	0.92 ± 0.00	41.0 ± 0.8
	Taylor $m = 1$	0.96 ± 0.00	**0.82 ± 0.01**	105.8 ± 0.7	2.36 ± 0.01	73.9 ± 0.8	1.65 ± 0.01	**0.91 ± 0.01**	40.7 ± 0.9
	Taylor $m = 2$	0.96 ± 0.00	0.80 ± 0.00	103.0 ± 1.5	2.31 ± 0.03	73.0 ± 0.8	1.64 ± 0.02	**0.91 ± 0.00**	41.5 ± 0.1
	AB $j = 2$	0.96 ± 0.00	0.81 ± 0.00	109.3 ± 2.1	2.43 ± 0.05	74.3 ± 1.8	1.65 ± 0.04	0.92 ± 0.01	40.7 ± 1.0
	AB $j = 3$	0.96 ± 0.00	**0.82 ± 0.00**	**102.6 ± 0.5**	**2.30 ± 0.02**	**71.2 ± 0.4**	**1.60 ± 0.01**	**0.91 ± 0.00**	41.2 ± 0.1

5.1 Predictive caching can match the base model's generation trajectory

In Figure 2, we visualize a representative single-atom coordinate trajectory under the base SemlaFlow model and SemlaFlow + TaylorSeer caching. To quantify agreement, we report the linear predictability error of sampling trajectories as a proxy for smoothness. Errors are largest in the early steps, which is consistent with the visible mismatch, but decrease thereafter. Across cache intervals D, the cached trajectory closely matches the base model in later steps. These observations indicate that predictive caching preserves the stepwise evolution of the generative process over most of the sampling horizon.

5.2 Caching allows for faster inference at iso-quality

Table 1 compares the default SemlaFlow configuration ($K = 100$ inference steps) to cached variants evaluated at multiple caching intervals and caching orders, as well as to a non-cached baseline with fewer steps. We omit the metrics novelty, uniqueness, and atom stability, on which all model variants consistently achieve 100%. We provide results on QM9 in Appendix A.

Compared to $K = 100$ solver steps, caching enables us to reduce the effective steps to 51 ($2\times$ throughput) while matching nearly all quality metrics. At a moderate cache interval ($D = 2$), we observe improvements in energy and strain in line with the quality improvements over the base model discussed in Liu et al. [2025a]. Quality remains comparable at 34 and 26 effective steps; at 26 steps, the cached model still matches the base model's low energy and achieves similar strain. Among caching configurations, higher-order predictors ($m = 2$, $j = 3$) perform best, and Adams–Bashforth caching consistently exceeds TaylorSeer. Besides higher optimized RMSD, the main drawback is a slight drop in validity, which is however offset by throughput gains of up to $4\times$.

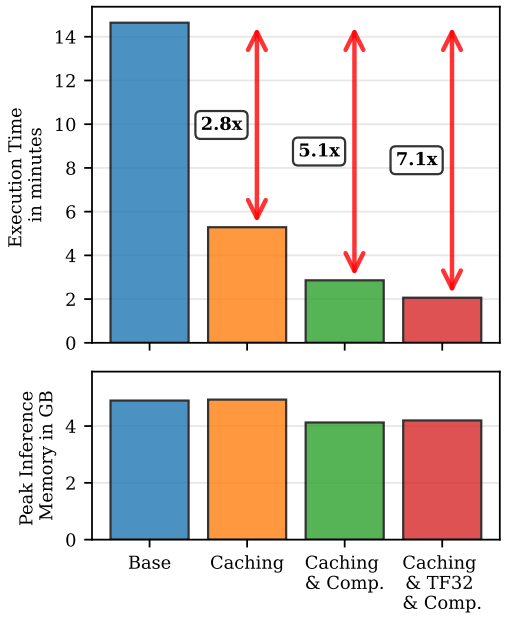

Figure 3: Inference time and memory overhead to sample 10,000 molecular geometries of various acceleration method combinations.

By contrast, naively reducing steps without caching results in significant quality degradation already at $K = 51$. Predictive feature caching therefore enables faster molecular geometry generation while preserving quality at practical operating points.

5.3 Caching is compatible with general inference acceleration methods

Predictive feature caching, as presented in this work, is complementary to lossless inference time optimizations. In Figure 3, we pair AB caching with graph compilation of the SemlaFlow backbone v_θ [Paszke et al., 2019]. The runtime branch that selects between evaluating $F(x_t)$ and using a forecast $F_{\text{pred},m}(x_{t+k})$ introduces control flow that would break whole-graph compilation. However, this can be easily avoided by compiling the backbone v_θ and keeping the selection logic outside the compiled region. We also enable TensorFloat-32 (TF32) matrix–multiply kernels to further increase throughput without affecting evaluation metrics.

In Figure 3, we report inference time and peak memory. Caching incurs a modest increase in peak memory due to maintaining $C(x_t)$, whereas compilation generally lowers peak memory. Caching alone yields $\sim$ 3x faster inference; combined with compilation and TF32, the speedup reaches up to 7x. This reduces the time to generate 10,000 molecules from $>$14 min to $\sim$2 min, with no significant loss in sampling quality.

6 Conclusion

In this paper, we address inference latency in molecular geometry generation by adapting a training-free predictive caching scheme to forecast intermediate hidden states during sampling for the molecular domain. Empirical evaluations quantify the speed–quality trade-off and demonstrate up to threefold reductions in wall-clock inference time while maintaining comparable conformer quality. More broadly, this work seeks to motivate systematic discussion of inference-time efficiency and to identify strategies for scaling to millions of samples.

Acknowledgements

We are grateful to Simon Langrieger and Gaspar Rochette for valuable feedback and discussions, which have greatly improved the quality of this paper, as well as to Begüm Çığ for her guidance on efficiency metrics.

References

Liang Wang, Chao Song, Zhiyuan Liu, Yu Rong, Qiang Liu, Shu Wu, and Liang Wang. Diffusion Models for Molecules: A Survey of Methods and Tasks, February 2025. URL `http://arxiv.org/abs/2502.09511`. arXiv:2502.09511 [cs].

Amira Alakhdar, Barnabas Poczos, and Newell Washburn. Diffusion Models in $\textit{De Novo}$ Drug Design. *Journal of Chemical Information and Modeling*, 64(19):7238–7256, October 2024.

Xiangru Tang, Howard Dai, Elizabeth Knight, Fang Wu, Yunyang Li, Tianxiao Li, and Mark Gerstein. A survey of generative AI for *de novo* drug design: new frontiers in molecule and protein generation. *Briefings in Bioinformatics*, 25(4), May 2024. ISSN 1467-5463, 1477-4054.

Simon Viet Johansson, Morteza Haghir Chehreghani, Ola Engkvist, and Alexander Schliep. De novo generated combinatorial library design. *Digital Discovery*, 3(1):122–135, 2024. doi: 10.1039/D3DD00095H. URL `https://pubs.rsc.org/en/content/articlelanding/2024/dd/d3dd00095h`. Publisher: Royal Society of Chemistry.

Lingling Shen, Jian Fang, Lulu Liu, Fei Yang, Jeremy L. Jenkins, Peter S. Kutchukian, and He Wang. Pocket Crafter: a 3D generative modeling based workflow for the rapid generation of hit molecules in drug discovery. *Journal of Cheminformatics*, 16(1):33, March 2024. ISSN 1758-2946. doi: 10.1186/s13321-024-00829-w.

Michał Koziarski, Mohammed Abukalam, Vedant Shah, Louis Vaillancourt, Doris Alexandra Schuetz, Moksh Jain, Almer van der Sloot, Mathieu Bourgey, Anne Marinier, and Yoshua Bengio. Towards DNA-Encoded Library Generation with GFlowNets. GEM workshop ICLR 2024, April 2024.

Yuyan Ni, Shikun Feng, Haohan Chi, Bowen Zheng, Huan-ang Gao, Wei-Ying Ma, Zhi-Ming Ma, and Yanyan Lan. Straight-Line Diffusion Model for Efficient 3D Molecular Generation, June 2025. URL `http://arxiv.org/abs/2503.02918`. arXiv:2503.02918 [cs].

Romain Lacombe and Neal Vaidya. Accelerating the Generation of Molecular Conformations with Progressive Distillation of Equivariant Latent Diffusion Models. Generative and Experimental Perspectives for Biomolecular Design Workshop at the 12th International Conference on Learning Representations, April 2024.

Minkai Xu, Alexander Powers, Ron Dror, Stefano Ermon, and Jure Leskovec. Geometric Latent Diffusion Models for 3D Molecule Generation. The Fortieth International Conference on Machine Learning, May 2023.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache Me if You Can: Accelerating Diffusion Models through Block Caching, 2023. URL https://arxiv.org/abs/2312.03209. Version Number: 2.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating Diffusion Models for Free, December 2023. URL http://arxiv.org/abs/2312.00858. arXiv:2312.00858 [cs].

Senmao Li, Taihang Hu, Joost van de Weijer, Fahad Shahbaz Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster Diffusion: Rethinking the Role of the Encoder for Diffusion Model Inference. 38th Conference on Neural Information Processing Systems, October 2024. URL http://arxiv.org/abs/2312.09608.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-Forward Caching in Diffusion Transformer Acceleration, July 2024. URL http://arxiv.org/abs/2407.01425. arXiv:2407.01425 [cs].

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. $$-DiT: A Training-Free Acceleration Method Tailored for Diffusion Transformers, June 2024. URL http://arxiv.org/abs/2406.01125. arXiv:2406.01125 [cs].

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From Reusing to Forecasting: Accelerating Diffusion Models with TaylorSeers. International Conference on Computer Vision, August 2025a.

Zichao Yu, Zhen Zou, Guojiang Shao, Chengwei Zhang, Shengze Xu, Jie Huang, Feng Zhao, Xiaodong Cun, and Wenyi Zhang. AB-Cache: Training-Free Acceleration of Diffusion Models via Adams-Bashforth Cached Feature Reuse, April 2025. URL http://arxiv.org/abs/2504.10540. arXiv:2504.10540 [stat].

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. Computer Vision and Pattern Recognition Conference, March 2025b. URL http://arxiv.org/abs/2411.19108.

Ziming Liu, Yifan Yang, Chengruidong Zhang, Yiqi Zhang, Lili Qiu, Yang You, and Yuqing Yang. Region-Adaptive Sampling for Diffusion Transformers, February 2025c. URL http://arxiv.org/abs/2502.10389. arXiv:2502.10389 [cs].

Wenzhang Sun, Qirui Hou, Donglin Di, Jiahui Yang, Yongjia Ma, and Jianxun Cui. UniCP: A Unified Caching and Pruning Framework for Efficient Video Generation, February 2025. URL http://arxiv.org/abs/2502.04393. arXiv:2502.04393 [cs].

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-Free Video Diffusion Model Acceleration with High Quality, March 2025. URL http://arxiv.org/abs/2410.19355. arXiv:2410.19355 [cs].

Zhihang Yuan, Hanling Zhang, Pu Lu, Xuefei Ning, Linfeng Zhang, Tianchen Zhao, Shengen Yan, Guohao Dai, and Yu Wang. DiTFastAttn: Attention Compression for Diffusion Transformer Models. 38th Conference on Neural Information Processing Systems, October 2024.

Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Learning a Continuous Representation of 3D Molecular Structures with Deep Generative Models. MLSB workshop at NeurIPS, November 2020.

Niklas W. A. Gebauer, Michael Gastegger, and Kristof T. Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules, January 2020. URL http://arxiv.org/abs/1906.00957. arXiv:1906.00957 [stat].

Youzhi Luo and Shuiwang Ji. An Autoregressive Flow Model for 3D Molecular Geometry Generation from Scratch. 42nd International Conference on Machine Learning, 2022.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. E(n) Equivariant Normalizing Flows. 35th Conference on Neural Information Processing Systems, January 2022.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D. International Conference on Machine Learning, June 2022.

Lei Huang, Hengtong Zhang, Tingyang Xu, and Ka-Chun Wong. MDM: Molecular Diffusion Model for 3D Molecule Generation, September 2022. URL http://arxiv.org/abs/2209.05710. arXiv:2209.05710 [cs].

Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Learning Joint 2D & 3D Diffusion Models for Complete Molecule Generation, June 2023. URL http://arxiv.org/abs/2305.12347. arXiv:2305.12347 [q-bio].

Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. MiDi: Mixed Graph and 3D Denoising Diffusion for Molecule Generation, June 2023. URL http://arxiv.org/abs/2302.09048. arXiv:2302.09048 [cs].

Bo Qiang, Yuxuan Song, Minkai Xu, Jingjing Gong, Bowen Gao, Hao Zhou, Weiying Ma, and Yanyan Lan. Coarse-to-Fine: a Hierarchical Diffusion Model for Molecule Generation in 3D. The Fortieth International Conference on Machine Learning, May 2023.

Alex Morehead and Jianlin Cheng. Geometry-Complete Diffusion for 3D Molecule Generation and Optimization. MLDD workshop @ ICLR, May 2024. doi: 10.48550/arXiv.2302.04313. URL http://arxiv.org/abs/2302.04313.

Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based Molecule Generation with Informative Prior Bridges, September 2022. URL http://arxiv.org/abs/2209.00865. arXiv:2209.00865 [cs].

Danny Reidenbach, Filipp Nikitin, Olexandr Isayev, and Saee Paliwal. Applications of Modular Co-Design for De Novo 3D Molecule Generation, May 2025. URL http://arxiv.org/abs/2505.18392. arXiv:2505.18392 [cs].

Haokai Hong, Wanyu Lin, and Kay Chen Tan. Accelerating 3D Molecule Generation via Jointly Geometric Optimal Transport. The Thirteenth International Conference on Learning Representations, March 2025.

Shikun Feng, Yuyan Ni, Yan Lu, Zhi-Ming Ma, Wei-Ying Ma, and Yanyan Lan. UniGEM: A Unified Approach to Generation and Property Prediction for Molecules. arXiv, April 2025. URL http://arxiv.org/abs/2410.10516. arXiv:2410.10516 [cs].

Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. SemlaFlow – Efficient 3D Molecular Generation with Latent Attention and Equivariant Flow Matching. 28th International Conference on Artificial Intelligence and Statistics, February 2025.

Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant Flow Matching with Hybrid Probability Transport. 37th Conference on Neural Information Processing Systems, December 2023.

Ian Dunn and David Ryan Koes. Mixed Continuous and Categorical Flow Matching for 3D De Novo Molecule Generation, April 2024. URL http://arxiv.org/abs/2404.19739. arXiv:2404.19739 [q-bio].

Chaitanya K. Joshi, Xiang Fu, Yi-Lun Liao, Vahe Gharakhanyan, Benjamin Kurt Miller, Anuroop Sriram, and Zachary W. Ulissi. All-atom Diffusion Transformers: Unified generative modelling of molecules and materials, May 2025. URL http://arxiv.org/abs/2503.03965. arXiv:2503.03965 [cs].

Sirine Ayadi, Leon Hetzel, Johanna Sommer, Fabian Theis, and Stephan Günnemann. Unified Guidance for Geometry-Conditioned Molecular Generation. 38th Conference on Neural Information Processing Systems, January 2025.

Mohamed Amine Ketata, Nicholas Gao, Johanna Sommer, Tom Wollschläger, and Stephan Günnemann. Lift Your Molecules: Molecular Graph Generation in Latent Euclidean Space. The Thirteenth International Conference on Learning Representations, June 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, February 2023. URL http://arxiv.org/abs/2210.02747. arXiv:2210.02747 [cs].

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. Transactions on Machine Learning Research, March 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow, September 2022. URL http://arxiv.org/abs/2209.03003. arXiv:2209.03003 [cs].

Xiaoliu Guan, Lielin Jiang, Hanqi Chen, Xu Zhang, Jiaxing Yan, Guanzhong Wang, Yi Liu, Zetao Zhang, and Yu Wu. Forecasting When to Forecast: Accelerating Diffusion Models with Confidence-Gated Taylor, August 2025. URL http://arxiv.org/abs/2508.02240. arXiv:2508.02240 [cs].

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, August 2014. ISSN 2052-4463. doi: 10.1038/sdata.2014.22. URL https://www.nature.com/articles/sdata201422. Publisher: Nature Publishing Group.

Simon Axelrod and Rafael Gomez-Bombarelli. GEOM: Energy-annotated molecular conformations for property prediction and molecular generation, February 2022. URL http://arxiv.org/abs/2006.05531. arXiv:2006.05531 [physics].

Tuan Le, Julian Cremer, Frank Noé, Djork-Arné Clevert, and Kristof Schütt. Navigating the Design Space of Equivariant Diffusion-Based Generative Models for De Novo 3D Molecule Generation, November 2023. URL http://arxiv.org/abs/2309.17296. arXiv:2309.17296 [cs].

Martin Buttenschoen, Yael Ziv, Garrett M. Morris, and Charlotte M. Deane. An evaluation of unconditional 3D molecular generation methods. GEM workshop ICLR 2025, May 2025.

Greg Landrum. Rdkit documentation, 2013.

Martin Buttenschoen, Garrett M. Morris, and Charlotte M. Deane. PoseBusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 15(9):3130–3139, 2024. ISSN 2041-6520, 2041-6539. doi: 10.1039/D3SC04185A. URL http://arxiv.org/abs/2308.05777.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019. URL http://arxiv.org/abs/1912.01703. arXiv:1912.01703 [cs].

A Results on QM9

Table 2: Evaluation of caching SemlaFlow trained on the QM9 Drugs dataset. We highlight in **bold** the best result per number of steps and underscore results that are better than or equal to the results of the base model at 50 steps.

Steps	Mode	Novelty ↓	Uniqueness ↑	Mol. Stability ↑	Valid (PRC)	Energy ↓	Energy p.A. ↓	Strain ↓	Strain p.A. ↓	Opt. RMSD ↓	Throughput ↑
50	Base	0.88 ± 0.00	0.95 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	40.9 ± 0.3	2.32 ± 0.02	15.4 ± 0.1	0.91 ± 0.01	0.23 ± 0.00	81.92 ± 0.14
26	Base	0.91 ± 0.00	0.95 ± 0.00	**0.99 ± 0.00**	**0.99 ± 0.00**	41.8 ± 0.3	2.37 ± 0.03	**15.7 ± 0.1**	**0.93 ± 0.00**	**0.23 ± 0.00**	140.61 ± 0.91
	Taylor	**0.90 ± 0.00**	**0.96 ± 0.00**	**0.99 ± 0.00**	**0.99 ± 0.00**	40.9 ± 0.5	2.33 ± 0.04	16.3 ± 0.2	0.97 ± 0.03	0.24 ± 0.00	144.27 ± 0.85
	AB	**0.90 ± 0.00**	**0.96 ± 0.00**	**0.99 ± 0.00**	**0.99 ± 0.00**	**40.9 ± 0.5**	**2.31 ± 0.02**	16.3 ± 0.2	0.96 ± 0.01	0.24 ± 0.00	142.15 ± 1.38
18	Base	0.91 ± 0.00	0.94 ± 0.00	**0.99 ± 0.00**	**0.99 ± 0.00**	42.5 ± 0.2	2.44 ± 0.05	**16.3 ± 0.2**	0.99 ± 0.05	**0.23 ± 0.00**	187.45 ± 1.54
	Taylor	**0.90 ± 0.00**	**0.96 ± 0.00**	**0.99 ± 0.00**	**0.99 ± 0.00**	**41.5 ± 0.2**	**2.37 ± 0.03**	16.4 ± 0.2	**0.98 ± 0.04**	0.24 ± 0.00	186.13 ± 1.49
	AB	0.91 ± 0.00	**0.96 ± 0.00**	**0.99 ± 0.00**	0.98 ± 0.00	42.5 ± 0.3	2.46 ± 0.08	16.7 ± 0.2	1.03 ± 0.07	0.24 ± 0.00	186.34 ± 2.35
14	Base	**0.91 ± 0.00**	0.94 ± 0.00	**0.99 ± 0.00**	**0.98 ± 0.00**	43.3 ± 0.1	**2.47 ± 0.02**	**17.1 ± 0.2**	**1.03 ± 0.03**	**0.23 ± 0.00**	220.53 ± 4.01
	Taylor	**0.91 ± 0.00**	**0.96 ± 0.00**	**0.99 ± 0.00**	0.97 ± 0.00	**43.2 ± 0.1**	**2.47 ± 0.02**	17.3 ± 0.1	1.04 ± 0.02	0.25 ± 0.00	228.10 ± 2.92
	AB	0.92 ± 0.00	**0.96 ± 0.00**	**0.99 ± 0.00**	0.97 ± 0.00	44.2 ± 0.6	2.55 ± 0.03	18.0 ± 0.4	1.10 ± 0.02	0.25 ± 0.00	217.92 ± 2.27

In Table 2, we present a subset of the evaluation from Table 1, conducted on SemlaFlow trained on QM9, a dataset comprising fewer samples and smaller molecules than GEOM. While GEOM is more relevant for practical applications, we include QM9 for completeness and to probe behavior in a simpler regime. We do not report atom stability, as all evaluated models saturate this metrics. Relative to Table 1, caching shows less quality degradation on QM9 in terms of valid molecules. At 26 effective steps, Adams–Bashforth ($j = 2$) caching performs best, but for even lower effective steps, TaylorSeer ($m = 1$) forecasting is superior overall. Across settings, cached models generally underperform the baseline on the strain metric, even when other metrics are matched.