Latent Flow Matching for Molecular Graph Generation

Anonymous authors
Paper under double-blind review

Abstract

Modern graph generative models typically operate directly in the discrete graph space, explicitly generating node and edge variables, which can become costly as graphs grow. In this paper, we perform generation explicitly on latent representations of entire graphs obtained from a pretrained Variational Autoencoder with high reconstruction fidelity. The generated representations, obtained through flow matching, are then decoded only at the final step. Across molecular benchmarks of increasing size, our approach achieves strong validity and FCD while offering a favorable quality-efficiency trade-off compared with state-of-the-art explicit graph generative models. One of the main advantages of this formulation is that the graph representation only needs to be learned once, after which the same one can be reused across multiple generative objectives without retraining. We demonstrate generation guided by molecular properties and further introduce validity-aware generation though a classifier learned directly in latent space. All code will be made available upon acceptance.

1 Introduction

Recent advances in graph generation have been largely driven by diffusion and flow-based models operating directly on explicit node and edge representations (Vignac et al., 2023; Qin et al., 2025; Roos et al., 2026). These methods model the graph distribution in its original discrete space and have achieved strong results across several graph generation benchmarks, but they suffer from having to manipulate large objects (full adjacency matrices and node embeddings) as well as being permutation sensitive by nature. An alternative line of work instead learns compact, permutation-invariant graph-level representations via autoencoder architectures (Winter et al., 2021; Krzakala et al., 2025; Bresson et al., 2026; Bengio et al., 2013), decoupling representation learning from distribution modeling (see Section 2 for details). One main drawbacks is that such architectures are known to yield poor generative performance without domain-specific knowledge being injected (Jin et al., 2018; Zahirnia et al., 2022).

This raises the question of whether an approach could leverage the best of both world, by moving distribution modeling entirely into such a continuous graph-level representation space; doing so should offer a competitive alternative to explicit graph generation in terms of sample quality, distributional fidelity, while retaining computational efficiency. This question is the central problem adressed in this work. Beyond unconditional generation, we ask whether the same latent representation can support controlled generation toward prescribed graph-level properties, and whether generation can be further biased toward latent regions that decode more reliably, improving chemical validity.

Our contributions are threefold. First, we train a flow-matching (Lipman et al., 2022) model to learn the latent distribution of the of a pretrained high-fidelity graph-level autoencoder. Second, we combine conditional flow matching with gradient-based guidance from differentiable property predictors, and demonstrate accurate control of logP and molecular weight while maintaining high validity and diversity on molecular datasets. Finally, we introduce an endpoint-based validity-aware flow matching formulation, based on Variational flow (Ei-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

jkelboom et al., 2024), that substantially improves chemical validity and characterize the resulting trade-off between validity and distributional fidelity.

2 Background and Related Work

Flow matching and generative modeling Generative modeling is the process of learning how to sample from an unknown data distribution q, so as to obtain new and varied data points. A usual process is to sample a point z from a known and simple prior p and learn a transformation T so that $T(z) \sim q$. In a VAE (Kipf & Welling, 2016), T is simply the decoder. More advanced methods, such as diffusion (Dhariwal & Nichol, 2021) and flow matching (Lipman et al., 2022) rely on applying a multi-step process, where each step removes noise and inject signal. In particular, flow matching learns this distribution by regressing a time-dependent vector field $v_\theta(z_t, t)$ that transports samples from a simple source distribution (e.g., a standard Gaussian) to $p(z)$ along a prescribed probability path, avoiding the iterative denoising schedules of diffusion models while retaining comparable sample quality. Conditional generation (Li et al., 2025) extends this formulation by conditioning q on desired properties of the generated object (i.e. a prompt). Flow matching enables this by conditioning the learned vector field directly Lipman et al. (2024).

Graph-Level Continuous Representations Let $\mathcal{G}$ denote the graph space, where a graph with n nodes is represented as an $n \times c_n$ matrix of node features and an adjacency tensor of shape $n \times n \times c_e$ representing edge classes. In contrast to node-level autoencoders Kipf & Welling (2016) (which encodes the graph as a permutation-equivariant $n \times d$ matrix), a graph-level encoder E maps a complete graph $G \in \mathcal{G}$ to a fixed-dimensional permutation-invariant representation $z = E(G) \in \mathbb{R}^d$, while a decoder D maps this representation back to graph space through $\hat{G} = D(z)$, with $\hat{G}$ an attempt at reconstructing G. Several recent architectures pursue this direction such as (Winter et al., 2021), GRALE (Krzakala et al., 2025), or GraViti (Bresson et al., 2026) differing primarily in how permutation invariance is enforced and in the fidelity of reconstruction on larger, more heterogeneous graphs. These approaches demonstrate that compact continuous representations can retain sufficient structural information for near-exact graph reconstruction on molecular datasets of different scales. In particular, it provides a variational autoencoder architecture (Kingma & Welling, 2014), as it structures the latent space explicitly with downstream generation in mind (Section 3). While efficient, one-shot VAE sampling is known to fail at generating consistently valid molecules (Kusner et al., 2017; Jin et al., 2018; Hu et al., 2023; Nguyen & Karolak, 2025), making it a poor generative model by default.

Generation in explicit graph space. Most recent graph generative models operate directly on node and edge variables, keeping the graph representation explicit throughout generation. DiGress (Vignac et al., 2023) applies discrete denoising diffusion to categorical graph representations, while CatFlow (Eijkelboom et al., 2024) and DeFoG (Qin et al., 2025) similarly formulate generation directly in graph space through categorical or discrete Flow Matching. More recently, Categorical Flow Maps (Roos et al., 2026) introduced continuous trajectories over categorical variables by evolving them in the probability simplex, enabling few-step generation. Despite this continuous formulation, the generative state still directly represents node and edge categories rather than a learned representation of the graph itself.

Generation through continuous representations. While latent generation approaches for graphs exist, they usually operate on node-level representations (Yang et al., 2024; Zhou et al., 2024; Siraudin & Morris, 2026; Xu et al., 2023) due to the only recent emergence of high-fidelity graph-level approaches. Graph-level approaches exist, starting with highly specialized VAEs leveraging domain-dependant constraints (Gómez-Bombarelli et al., 2018; Jin et al., 2018) offering high validity, but with capped distributional fidelity (Vignac et al., 2023). Building on those graph-level latents, ChemFlow (Wei et al., 2024) has shown interesting latent navigation, without being a generative model, while NGG (Evdaimon et al., 2024) employs a small variational graph autoencoder on general graphs based on structure conditioning, limiting its applicability to more complex data.

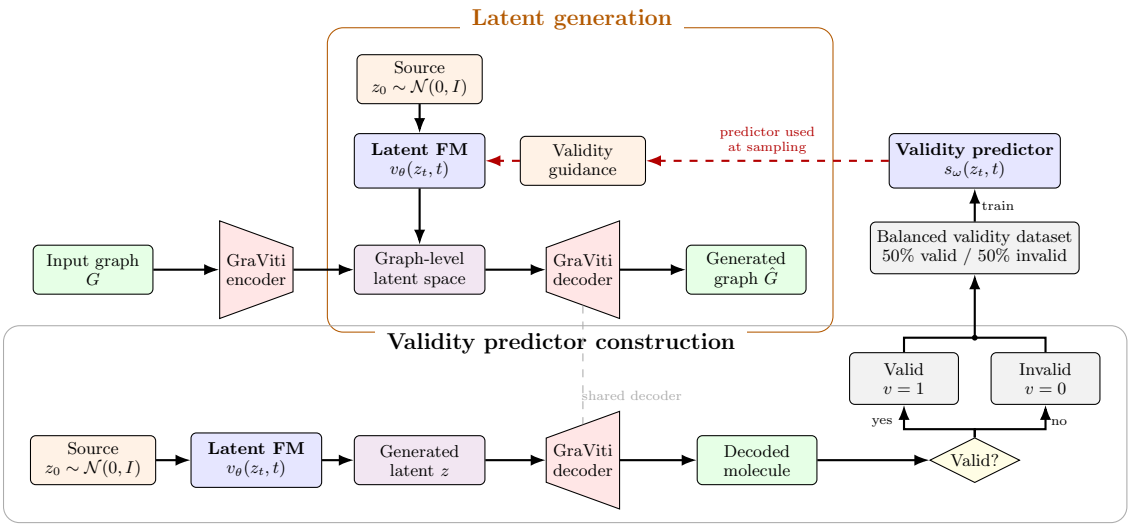


Figure 1: Overview of latent graph generation and validity-guided sampling. Top: latent flow matching; Bottom: construction of the validity dataset and validity predictor training.

3 Method

Latent Flow Matching We formulate graph generation as distribution learning in a graph-level continuous representation space. Representations are obtained from a pretrained GraViti-based variational autoencoder (Bresson et al., 2026), since it is freely available and trainable and achieves high reconstruction quality. The encoder and decoder are considered as pretrained, and remain fixed, decoupling graph representation learning from generative modeling.

Let $E : \mathcal{G} \to \mathbb{R}^d$ denote the graph encoder of our pretrained VAE and $z = E(G) \in \mathbb{R}^d$ the latent representation of a graph G. We learn the latent graph distribution using Flow Matching (Lipman et al., 2022), transporting an isotropic Gaussian source distribution $z_0 \sim \mathcal{N}(0, I) \in \mathbb{R}^d$ toward latent graph embeddings $z_1 \sim p_{\text{data}} \in \mathbb{R}^d$, where p_{data} is the empirical distribution of encoder outputs obtained by applying E to the training graphs. We adopt the standard Flow Matching formulation, detailed in Appendix E. To improve the coupling between source and target samples, we employ mini-batch optimal transport (Tong et al., 2024), which promotes shorter and straighter transport trajectories while preserving the source and target marginals.

In order to avoid data leakage from the VAE's training, we adopt the same training, validation and test splits that are provided with GraViti. Therefore, both our model and the pretrained VAEs are trained on the same molecules exactly. We then learn a neural flow model over the distribution of graph representations, such that the complete generative trajectory is modeled in continuous latent space and only the final representation is decoded back to graph space. At generation time, the learned field is integrated from $t = 0$ to $t = 1$ and the resulting latent endpoint is decoded into a graph only after the last step.

Validity Guidance We further exploit the differentiability of the latent dynamics to promote molecule validity. After an unconditional neural flow model is trained, a time-dependent classifier s_ω is trained on valid and invalid latent states. The classifier is time-dependent because, given an intermediate latent state z_t and its flow time t, it predicts whether the corresponding sample is expected to decode into a valid graph at the end of the trajectory. To train this prediction throughout the flow, intermediate states are constructed along linear trajectories between noise samples and valid or invalid endpoint latents, with each state inheriting the validity label of its endpoint. Positive examples are obtained from encoded valid training graphs, while negative examples are generated by the latent Flow Matching model and retained when they decode into invalid graphs. The construction of

this dataset is critical to obtaining informative guidance: rather than relying on easily separable negatives far from the graph latent distribution, we use hard negatives lying in regions actually explored by the generator. This encourages the classifier to learn the validity boundary relevant to the sampling trajectories. Following the principle of classifier guidance (Dhariwal & Nichol, 2021), sampling can be steered according to

$$\tilde{v}_\theta(z_t, t) = v_\theta(z_t, t) + \mathbb{1}[t \geq t_{\text{start}}]\lambda_{\text{val}}\nabla_{z_t} \log p_\omega(\text{valid} \mid z_t, t). \tag{1}$$

Where p_w is the validity probability obtained from s_w and λ_{val} is a parameter that controls the steering intensity. For stability and diversity, we restrict guidance to the later part of the trajectory (after t_{start}), where samples approach the learned graph distribution.

Property-Controlled Generation A desirable property of generative models is the ability to be prompted by properties of the generated molecule. We condition the flow matching field on a target property vector c, yielding $v_\theta(z_t, t, c)$, and complement this conditioning with a differentiable property predictor $r_\omega(z_t, t)$ trained on intermediate states of the generative trajectories (using a similar process to that of the validity classifier). For a target c, the predictor provides an adaptive correction during sampling,

$$\tilde{v}_\theta(z_t, t, c) = v_\theta(z_t, t, c) - \mathbb{1}[t \geq t_{\text{start}}]\lambda_{\text{prop}} \nabla_{z_t} \mathcal{L}_{\text{prop}}\big(r_\omega(z_t, t), c\big), \tag{2}$$

where $\mathcal{L}_{\text{prop}}$ is the squared difference between the predicted property and the target value c, and where guidance is applied only once t exceeds a threshold t_{start}, since r_ω's gradient is unreliable while z_t remains close to pure noise. Conditioning therefore shapes the global transport toward the requested property, while gradient guidance refines individual trajectories once sufficient structure has emerged. Property control remains entirely within the graph-level latent space and requires no modification of the graph decoder.

Validity-Aware Endpoint Generation Validity guidance modifies a pretrained generative field at sampling time. We also investigate whether validity can be incorporated directly into the generative distribution. Building on the endpoint formulation of Variational Flow Matching (Eijkelboom et al., 2024), we model a conditional distribution over final graph representations,

$$q_\theta(z_1 \mid z_t, t) = \mathcal{N}\big(\mu_\theta(z_t, t), \text{diag}\big(\sigma^2_\theta(z_t, t)\big)\big)\,. \tag{3}$$

Let $a_\omega(z) = \sigma(s_\omega(z, 1))$ denote the validity score produced by a frozen classifier. When interpreted as an estimate of $P(V = 1 \mid z)$, where V denotes graph validity, the joint density of an endpoint and the validity event factorize as

$$p_{\theta,\omega}(z_1, V = 1 \mid z_t, t) = q_\theta(z_1 \mid z_t, t)\, a_\omega(z_1). \tag{4}$$

Importantly, a_ω is a conditional validity probability and not a density over latent space. The probability that an endpoint sampled from the model is valid is therefore

$$P_{\theta,\omega}(V = 1 \mid z_t, t) = \mathbb{E}_{z_1 \sim q_\theta}\left[a_\omega(z_1)\right] \tag{5}$$

Motivated by this factorization, we train the endpoint distribution with $\mathcal{L}_{\text{end}} = \mathcal{L}_{\text{NLL}} + \lambda_{\text{val}}\mathcal{L}_{\text{val}}$, where $\mathcal{L}_{\text{NLL}} = \mathbb{E}_{z_1 \sim p_{\text{data}}}\left[-\log q_\theta(z_1 \mid z_t, t)\right]$ and $\mathcal{L}_{\text{val}} = \mathbb{E}_{\tilde{z}_1 \sim q_\theta}\left[-\log a_\omega(\tilde{z}_1)\right]$. The first term preserves likelihood on observed graph representations, while the second explicitly moves probability mass toward endpoints predicted to decode into valid graphs. Since $-\log$ is convex, Jensen's inequality gives

$$\mathcal{L}_{\text{val}} \geq -\log P_{\theta,\omega}(V = 1 \mid z_t, t), \tag{6}$$

providing a direct probabilistic interpretation of the validity objective. The samples $\tilde{z}_1$ are reparameterized through q_θ, allowing validity gradients to update the endpoint distribution.

At inference, the endpoint distribution can be exploited in two ways. Its predicted mean provides a deterministic endpoint estimate, while stochastic sampling allows several plausible endpoints to be considered at each step. We draw K candidates from $q_\theta(z_1 \mid z_t, t)$ and favor those with higher predicted validity,

$$\hat{z}_1 = \sum_{k=1}^{K} w_k z_1^{(k)}, \qquad w_k \propto \exp\Big(\beta s_\omega(z_1^{(k)}, 1)\Big)\,. \tag{7}$$

This reweighting exploits the uncertainty of the learned endpoint distribution while favoring trajectories toward latent regions with higher predicted validity.

Table 1: Statistics of the used datasets. Numbers of atoms and bonds are reported on average. Last column is the dimension of the latent representation.

Name	# Graph	# atom	# bonds	Max Size	Atom types	Emb. Dim.
QM9	130k	8.79	9.4	9	4	64
PubChem16	12.89M	13.87	14.19	16	31	256
PubChem32	74.34M	21.75	21.15	32	31	512

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our approach on QM9 (Ramakrishnan et al., 2014) and PubChem (Kim et al., 2016), two molecular graph datasets with substantially different scales and structural complexity. For PubChem, we use two subsets clipped at a maximum size of 16 and 32 heavy atoms respectively. All datasets have 4 edge types, corresponding to simple, double, triple bonds, and one extra class for "no bond". Statistics are described in Table 1.

Baselines. We compare latent generation against explicit graph generative models: the diffusion-based DiGress (Vignac et al., 2023), a the flow-based Categorical Flow Maps (Roos et al., 2026). We additionally report Gaussian sampling from the VAE prior as a latent-space baseline, allowing us to quantify the benefit of learning the latent graph distribution with flow matching. All models are trained and evaluated on the same splits.

Evaluation protocol. For each configuration, we perform 20 independent generation runs of 1,024 molecules, for a total of 20,480 generated samples. Implementation details and hyperparameters are provided in appendix I.

4.2 Unconditional Graph Generation and Validity Guidance

We first evaluate whether graph generation can be performed effectively by modeling the distribution of compact graph-level continuous representations, rather than explicitly generating discrete node and edge variables. This comparison is designed to evaluate not only generation quality, but also the computational trade-offs associated with the different generative representations. DiGress operates explicitly in discrete graph space through a long denoising trajectory, while CFM is specifically designed for efficient few-step categorical generation. We report the standard metrics of validity, uniqueness, novelty, Fréchet Chemnet Distance (Preuer et al., 2018) to assess distributional fidelity, and end-to-end sampling time. We additionally report computational efficiency to compare latent and explicit graph generation beyond sample quality. Finally, we compute a metric, which we call **VUN/T**, which is computed as the quadruple product of validity, uniqueness, novelty and batch size, divided by the generation time; it corresponds to the number of valid, unique and novel molecules obtained per second, and serves as a summary of the generation efficiency. All results are reported in Table 2.

Explicit methods dominate on QM9, composed of very small molecules, although all methods achieve high performance on this dataset. Already at this scale, a key advantage of latent flow matching emerges: total generation time grows affinely with the number of integration steps for both approaches, but with a much shallower slope for latent FM, and the dominant cost is the single, fixed final decoding step. Explicit approaches, by contrast, pay a substantially steeper per-step cost, as every integration step operates on the full graph representation.

On PubChem16, our model shows a clear advantage in distributional fidelity, achieving substantially better FCD than both CFM variants, even at only 20 NFE, and outperforming DiGress by a significant margin. Notably, increasing the number of integration steps has only a marginal effect on our model's fidelity, in contrast to explicit methods. DiGress attains the highest validity rate, but only in its 500-step regime, which requires 70 seconds

Table 2: Unconditional generation results on our three benchmarks. Validity, novelty and uniqueness are reported as mean ± standard deviation over 20 runs of 1,024 molecules. FCD is computed over the generated set. Time denotes end-to-end generation of 1,024 molecules and includes the VAE decoding for our method. NFE stands for number of function evaluations (1 per step for Euler, 2 per step for midpoint and Heun solver). The numerical solvers are detailed in Appendix E.3. Our approaches are Latent FM, Latent FM with validity guidance (VG) and Validity-Aware VFM (VA-VFM). Qualitative examples of pc32 generation are provided in Appendix F.

Method	Sampler	NFE	Validity ↑	Uniq. ↑	Novelty ↑	FCD ↓	Time (s) ↓	VUN/T ↑
QM9								
DiGress	Diffusion	2	47.3 ± 1.4	97.5 ± 0.9	N/A	3.420	0.324 ± 0.081	1457.5
DiGress	Diffusion	5	76.7 ± 1.7	99.5 ± 0.3	N/A	1.150	0.633 ± 0.005	1234.6
DiGress	Diffusion	20	97.8 ± 0.4	$\mathbf{99.7} \pm 0.2$	N/A	0.087	2.275 ± 0.030	438.9
DiGress	Diffusion	500	$\mathbf{99.9} \pm 0.1$	99.6 ± 0.2	N/A	**0.061**	54.801 ± 0.13	18.6
CFM-CSD	CSD	1	94.6 ± 0.6	98.6 ± 0.6	N/A	1.873	0.391 ± 0.005	2442.8
CFM-CSD	CSD	2	96.5 ± 0.4	$\mathbf{99.7} \pm 0.1$	N/A	0.280	0.446 ± 0.002	2209.0
CFM-CSD	CSD	5	98.8 ± 0.3	$\mathbf{99.7} \pm 0.2$	N/A	0.111	0.632 ± 0.002	1596.0
CFM-CSD	CSD	20	99.6 ± 0.2	99.6 ± 0.2	N/A	0.082	1.520 ± 0.002	668.3
CFM-CSD	CSD	500	99.7 ± 0.1	99.5 ± 0.1	N/A	0.069	30.129 ± 0.194	33.71
CFM-ECLD	ECLD	1	97.7 ± 0.4	97.3 ± 0.5	N/A	1.791	0.401 ± 0.003	2427.5
CFM-ECLD	ECLD	2	98.2 ± 0.4	99.6 ± 0.2	N/A	0.480	0.447 ± 0.002	2240.6
CFM-ECLD	ECLD	5	99.2 ± 0.3	99.6 ± 0.2	N/A	0.327	0.626 ± 0.003	1616.2
CFM-ECLD	ECLD	20	99.5 ± 0.3	99.6 ± 0.2	N/A	0.342	1.522 ± 0.002	666.8
CFM-ECLD	ECLD	500	99.5 ± 0.3	99.5 ± 0.2	N/A	0.305	29.955 ± 0.031	33.84
VAE Sampling	N/A	N/A	93.1 ± 0.7	98.45 ± 0.2	N/A	2.88	N/A	N/A
Latent FM	Heun	2	96.7 ± 0.6	99.6 ± 0.1	N/A	0.390	1.840 ± 0.005	536.0
Latent FM	Heun	5	97.9 ± 0.4	$\mathbf{99.7} \pm 0.1$	N/A	0.156	1.836 ± 0.007	544.4
Latent FM	Heun	20	98.2 ± 0.4	$\mathbf{99.7} \pm 0.2$	N/A	0.110	1.870 ± 0.014	536.1
Latent FM	Heun	500	98.4 ± 0.3	$\mathbf{99.7} \pm 0.2$	N/A	0.127	3.095 ± 0.019	324.6
Lat. FM + VG	Midpoint	20	99.5 ± 0.2	$\mathbf{99.7} \pm 0.1$	N/A	0.110	1.910 ± 0.023	531.8
VA-VFM	Euler	20	99.7 ± 0.1	99.3 ± 0.2	N/A	0.760	1.880 ± 0.021	539.3
PubChem16								
DiGress	Diffusion	2	29.1 ± 1.9	99.4 ± 0.5	77.1 ± 1.9	10.44	0.403 ± 0.044	566.7
DiGress	Diffusion	5	58.7 ± 1.8	99.9 ± 0.1	82.2 ± 1.3	6.060	0.817 ± 0.029	604.2
DiGress	Diffusion	20	94.9 ± 0.7	100.0 ± 0.0	83.1 ± 1.2	0.945	2.928 ± 0.026	275.8
DiGress	Diffusion	500	$\mathbf{99.6} \pm 0.2$	$\mathbf{100.0} \pm 0.0$	81.6 ± 1.1	0.356	70.637 ± 0.058	11.8
CFM-CSD	CSD	1	87.1 ± 1.2	99.6 ± 0.3	75.7 ± 1.5	6.978	0.510 ± 0.002	1318.6
CFM-CSD	CSD	2	84.8 ± 1.2	$\mathbf{100.0} \pm 0.1$	87.8 ± 1.1	2.901	0.670 ± 0.002	1137.9
CFM-CSD	CSD	5	92.7 ± 1.0	$\mathbf{100.0} \pm 0.0$	88.6 ± 1.2	1.768	1.151 ± 0.003	730.7
CFM-CSD	CSD	20	97.0 ± 0.7	$\mathbf{100.0} \pm 0.0$	$\mathbf{89.2} \pm 1.0$	1.155	3.562 ± 0.004	248.7
CFM-CSD	CSD	500	98.3 ± 0.4	99.9 ± 0.1	87.8 ± 1.1	0.70	82.362 ± 0.111	10.71
CFM-ECLD	ECLD	1	95.9 ± 0.7	98.7 ± 0.4	64.7 ± 1.3	7.034	0.511 ± 0.002	1227.2
CFM-ECLD	ECLD	2	93.7 ± 0.6	$\mathbf{100.0} \pm 0.1$	86.0 ± 1.0	2.949	0.670 ± 0.002	1231.6
CFM-ECLD	ECLD	5	96.1 ± 0.5	$\mathbf{100.0} \pm 0.0$	88.9 ± 1.1	2.339	1.153 ± 0.002	758.8
CFM-ECLD	ECLD	20	97.4 ± 0.5	$\mathbf{100.0} \pm 0.0$	89.0 ± 1.1	2.083	3.554 ± 0.004	249.8
CFM-ECLD	ECLD	500	97.8 ± 0.4	99.9 ± 0.1	88.5 ± 1.1	1.462	82.361 ± 0.105	10.75
VAE Sampling	N/A	N/A	67.3 ± 1.5	100.0 ± 0.0	91.7 ± 1.0	4.201	N/A	N/A
Latent FM	Heun	2	94.7 ± 0.6	99.8 ± 0.1	52.8 ± 1.6	2.33	3.498 ± 0.030	146.1
Latent FM	Heun	5	95.6 ± 0.5	100.0 ± 0.0	79.0 ± 1.4	0.409	3.757 ± 0.027	205.8
Latent FM	Heun	20	97.3 ± 0.5	$\mathbf{100.0} \pm 0.0$	81.2 ± 1.3	0.190	4.260 ± 0.039	189.9
Latent FM	Heun	500	97.7 ± 0.5	$\mathbf{100.0} \pm 0.0$	80.4 ± 1.5	**0.171**	10.274 ± 0.056	78.3
Lat. FM + VG	Midpoint	20	97.8 ± 0.4	$\mathbf{100.0} \pm 0.0$	81.0 ± 1.1	0.210	4.430 ± 0.040	183.1
VA-VFM	Euler	20	98.6 ± 0.3	99.9 ± 0.1	82.1 ± 1.2	1.370	4.420 ± 0.029	187.4
PubChem32								
DiGress	Diffusion	20	89.2 ± 0.7	$\mathbf{100.0} \pm 0.0$	93.0 ± 0.8	3.24	7.922 ± 0.133	107.23
DiGress	Diffusion	500	99.0 ± 0.3	$\mathbf{100.0} \pm 0.0$	95.1 ± 0.5	1.17	193.77 ± 0.054	3.406
VAE Sampling	N/A	N/A	11.1 ± 0.1	100.0 ± 0.0	92.61 ± 0.02	11.33	N/A	N/A
Latent FM	Heun	2	78.6 ± 0.8	93.7 ± 0.9	36.7 ± 1.4	13.52	4.764 ± 0.031	58.1
Latent FM	Heun	20	74.5 ± 0.6	$\mathbf{100.0} \pm 0.0$	91.1 ± 1.1	0.868	9.425 ± 0.123	73.8
Latent FM	Heun	500	74.4 ± 1.9	$\mathbf{100.0} \pm 0.0$	91.2 ± 1.2	**0.856**	17.927 ± 0.126	38.8

per batch, compared to 4 seconds for our model at 20 NFE (or 10 seconds at 500 NFE), while our model still achieves significantly lower (better) FCD in both regimes, with a **VUN/T** up to 17 times higher.

Table 3: Single-GPU training cost on QM9 and PubChem16. Times for the proposed methods assume a fixed pretrained graph representation and therefore exclude the one-time encoder-decoder pretraining cost.

Method / component	**QM9**	**PubChem16**
DiGress	7 h 34 min	16 h 55 min
CFM-CSD	63 h 49 min	73 h 21 min
CFM-ECLD	70 h 53 min	71 h 40 min
Latent FM	27 min	3 h 50 min
Validity dataset construction	2 h 06 min	8 h 05 min
Validity predictor	1 min 28 s	1 min 12 s
Validity-aware VFM training	15 min 24 s	12 h 08 min
Latent FM + validity guidance	2 h 35 min	11 h 57 min
Complete latent pipeline	2 h 50 min	24 h 05 min

On Pubchem32, we did not train CFM due to its very large training cost (73 hours on PubChem16, which is 7 times smaller). We therefore compare to DiGress only. First of all we notice that, like for the other datasets, our model has significantly improved the generation performance with regards to the base VAE, improving validity 7-fold. Nonetheless, it does not manage to fully compensate for this, stagnating around 74.5% validity while Digress reaches 99% with full 500 steps.

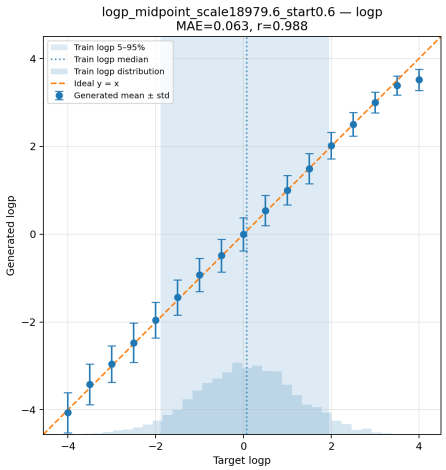
Validity guidance further improves the latent FM by increasing chemical validity while preserving its distributional fidelity. The validity-aware VFM increases validity on QM9 and PubChem16, but at the cost of significant FCD; it should therefore be viewed as a specialized alternative for applications in which validity is the primary objective. It reaches the highest validity among our latent variants, but at the cost of a larger FCD, making the validity-fidelity trade-off explicit. On PubChem32, validity guidance fails at improving the validity on that dataset, which we attribute to the complexity of the latent space for that dataset relative to the small size of the validity predictor's training set (800,000 samples).

Training efficiency. The computational profiles also differ substantially at training time, as reported in Tables 3. CFM provides the fastest inference but requires the longest training among the evaluated approaches. DiGress has a more moderate training cost, while its strongest generation results rely on long sampling trajectories. In contrast, latent flow matching is particularly inexpensive to train, as it exploits a pre-traiend VAE, with a training cost on QM9 that is roughly two orders of magnitude below that of CFM.

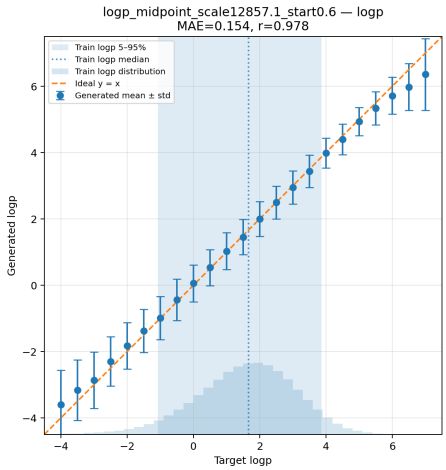
Even when validity dataset construction, classifier training, and the validity-aware endpoint model are included, the complete latent pipeline remains substantially cheaper to train than CFM on QM9. A similar trend is observed on PubChem16, where the latent FM alone remains considerably less expensive to train than the explicit categorical flow models (Table 3). These comparisons exclude the one-time cost of learning the graph encoder-decoder used by the latent models. However, this representation is shared across downstream tasks rather than being trained specifically for each generative objective. Once learned, the same continuous graph space is reused for unconditional generation, validity guidance, validity-aware generation, and property control. Even if trained specifically for latent flow matching purposes, the cost of representation learning can therefore be amortized across multiple tasks, which is a central advantage of decoupling graph representation learning from generative modeling.

4.3 Property-Guided Generation

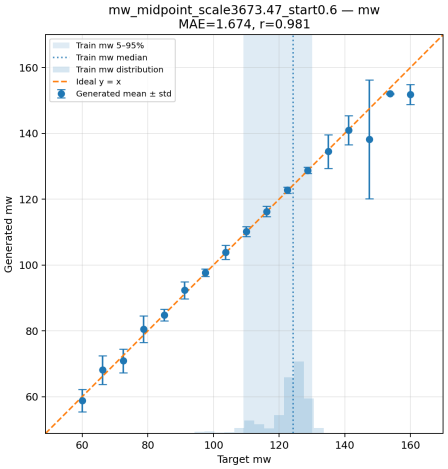
We study independent control of two complementary molecular properties, molecular weight (MW) and logP, prescribing one property at a time. MW mainly reflects molecular size and atomic composition, whereas logP depends more strongly on molecular structure and functional-group arrangement. This allows us to assess whether a single continuous latent



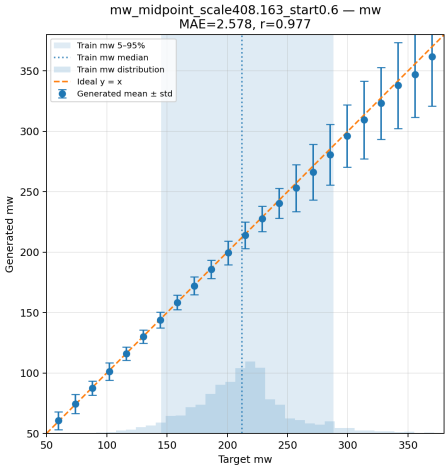
(a) QM9 - logP.



(b) PubChem16 - logP.



(c) QM9 - molecular weight.



(d) PubChem16 - molecular weight.

Figure 2: Independent property-controlled generation on QM9 and PubChem16. Points show the mean property of generated molecules for each requested target, with error bars indicating one standard deviation. The dashed line corresponds to ideal control, while the shaded region indicates the 5-95% range of the training distribution.

representation can support distinct forms of molecular control rather than a single property class.

Independent property control. For each property, we condition the latent Flow Matching model on the target and optionally refine sampling with gradient-based guidance from a property predictor. Targets cover both high-density and more extreme regions of the training distribution, allowing us to assess controllability beyond frequent property values.

Figure 2 compares requested and generated properties for the best guided configurations. Across QM9 and PubChem16, generated means closely track the targets over a broad range, showing that the same latent representation supports both unconditional generation and property control. Qualitative examples are provided in Appendix G.

Although conditioning alone already provides effective property control, gradient-based guidance proves substantially more effective at improving target fidelity, with the largest gains observed for logP and consistent improvements for MW (Table 4).

Table 4: Effect of gradient-based property guidance on target fidelity. We compare conditional flow matching (CFM) with CFM augmented with property-predictor guidance. Lower MAE and higher Pearson correlation r indicate better property control.

Dataset	Property	Conditional FM		+ Guidance	
		MAE ↓	r ↑	MAE ↓	r ↑
QM9	logP	0.100	0.978	**0.063**	**0.988**
QM9	MW	1.705	0.979	**1.674**	**0.981**
PubChem16	logP	0.282	0.966	**0.154**	**0.978**
PubChem16	MW	3.040	0.976	**2.578**	**0.977**

Influence of the latent representation. Both deterministic and variational GraViti representations support effective property control, although the variational latent space consistently yields more accurate target matching (Appendix H).

Validity under property control. Target fidelity alone is insufficient to characterize controlled generation: a model may reach increasingly extreme property values by moving toward latent regions that decode poorly. We therefore distinguish between global and local validity. Global validity evaluates the conditional generation regime in which target properties are sampled according to their empirical training distribution. It therefore gives greater weight to targets lying in high-density regions of the property space. Local validity instead evaluates validity separately at each prescribed target and reveals how generation quality evolves when moving toward rare or out-of-distribution property values.

Under global conditioning, property guidance preserves high validity and uniqueness on both datasets, remaining close to the unconditional regime on QM9 and above 0.92 validity with maximal uniqueness on PubChem16. Detailed results are reported in Appendix D.

Global validity, however, can conceal substantial variations across the property range. Since targets are sampled according to the empirical training distribution, the resulting average is dominated by high-density regions and provides limited information about the behavior of the model near the boundaries of, or beyond, the training distribution. We therefore additionally report *local validity*, computed independently for each prescribed target value. Local validity and uniqueness remain high within the training distribution, while target-wise results in Appendix B show that validity degrades primarily for increasingly rare or out-of-distribution targets. These results also qualitatively indicate that our models remain reasonably robust beyond the training range.

5 Conclusion

In this paper, we introduce a method for molecule generation using a flow matching algorithm in the latent space of an autoencoder. We show that our approach performs on par with explicit graph generation approaches, with the most significant tradeoff being that our approach sacrifices some validity, but at the benefit of improved distributional fidelity and a massive gain in time, especially as graphs get larger. We show that despite the decreased fidelity, the time required to generate a given amount of valid molecule is favorable, and leverage our method to provide guided generation on chemical properties, yielding high property fidelity even out-of-distribution.

This work offers a promising approach to graph and molecule generation processes that scale better, avoiding the high cost of in-graph-space explicit propagation.

AI use statement

The manuscript was written by the authors. Generative AI tools were used for limited assistance with restructuring, LaTeX formatting, and occasional language editing. They were also used as coding and debugging assistants and to provide feedback on experimental design and implementation details.

All AI-assisted text, code, methodological suggestions, and interpretations were reviewed and verified by the authors. Experimental results were obtained from the authors' implementations and experiments, and all reported results and scientific claims were independently checked by the authors.

The authors take full responsibility for the final content of this work, including any text, code, or other artifacts produced with the assistance of generative AI tools.

Reproducibility statement

We provide detailed model architectures, training hyperparameters, sampling settings, and evaluation protocols in the appendix. All reported metrics are computed using the procedures described in the experimental section, with multiple independent generation runs when applicable. The used VAE and datasets are open source. Code will be made available upon acceptance

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

Roman Bresson, Konstantinos Divriotis, Johannes F. Lutzeyer, Iakovos Evdaimon, and Michalis Vazirgiannis. GraViti: Graph-level variational autoencoders with relaxed permutation invariance. *arXiv preprint arXiv:2605.16668*, 2026. URL https://arxiv.org/abs/2605.16668.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.04843.

Iakovos Evdaimon, Giannis Nikolentzos, Christos Xypolopoulos, Ahmed Kammoun, Michail Chatzianastasis, Hadi Abdine, and Michalis Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models. *arXiv preprint arXiv:2403.01535*, 2024.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018. doi: 10.1021/acscentsci.7b00572.

Chao Hu, Song Li, Chenxing Yang, Jun Chen, Yi Xiong, Guisheng Fan, Hao Liu, and Liang Hong. ScaffoldGVAE: scaffold generation and hopping of drug molecules via a variational autoencoder based on multi-view graph neural networks. *J. Cheminform.*, 15 (1):91, October 2023.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2323–2332. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/jin18a.html.

Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. Pubchem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 01 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv951. URL https://doi.org/10.1093/nar/gkv951.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv:1611.07308 [cs, stat]*, November 2016. URL http://arxiv.org/abs/1611.07308. arXiv: 1611.07308.

Paul Krzakala, Gabriel Melo, Charlotte Laclau, Florence d'Alché Buc, and Rémi Flamary. The quest for the GRAph level autoencoder (GRALE). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1945–1954. JMLR.org, 2017.

Xiang Li, Rongrong Wang, and Qing Qu. Towards understanding the mechanisms of classifier-free guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=bRAm7A02Qm.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2210.02747.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Trieu Nguyen and Aleksandra Karolak. Transformer graph variational autoencoder for generative molecular design. *Biophysical Journal*, 124(22):3867–3875, 2025. ISSN 0006-3495. doi: https://doi.org/10.1016/j.bpj.2025.01.022. URL https://www.sciencedirect.com/science/article/pii/S0006349525000359.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet chemnet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, 2018. doi: 10.1021/acs.jcim.8b00234.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. In *International Conference on Machine Learning (ICML)*, 2025.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1: 140022, 2014. doi: 10.1038/sdata.2014.22.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, Ismail Ilkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. *arXiv preprint arXiv:2602.12233*, 2026. URL https://arxiv.org/abs/2602.12233.

Antoine Siraudin and Christopher Morris. Principled latent diffusion for graphs via laplacian autoencoders. *arXiv preprint arXiv:2601.13780*, 2026. URL https://arxiv.org/abs/2601.13780. Preprint.

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*, 2024.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.14734.

Guanghao Wei, Yining Huang, Chenru Duan, Yue Song, and Yuanqi Du. Navigating chemical space with latent flows. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.03987.

Robin Winter, Frank Noe, and Djork-Arné Clevert. Permutation-invariant variational autoencoder for graph-level representation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9559–9573. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f3d7d38d24b740c95da2b03dc3a2333-Paper.pdf.

Minkai Xu, Alexander Powers, Ron Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*. PMLR, 2023.

Ling Yang, Zhilin Huang, Zhilong Zhang, Zhongyi Liu, Shenda Hong, Wentao Zhang, Wenming Yang, Bin Cui, and Luxia Zhang. Graphusion: Latent diffusion for graph generation. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6358–6369, 2024. doi: 10.1109/TKDE.2024.3389783.

Kiarash Zahirnia, Oliver Schulte, Parmis Naddaf, and Ke Li. Micro and macro level graph modeling for graph variational auto-encoders. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30347–30361. Curran Associates, Inc., 2022. doi: 10.52202/068431-2200. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c400474e8a36d0812fdee52739288b12-Paper-Conference.pdf.

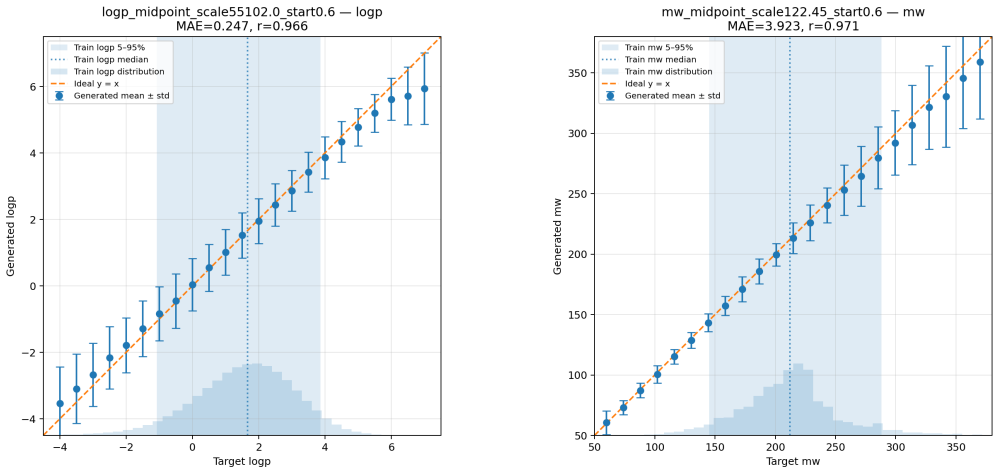
Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lvibangnAs.

A Influence of the Latent Representation on Property Control

To further assess the influence of the graph-level representation on property-controlled generation, we compare the deterministic and variational GraViti latent spaces on PubChem16. In both cases, we use the same dataset, controlled properties, and property-guidance protocol, changing only the latent representation in which the generative dynamics are learned.

Figure 3 reports the target-response curves obtained in the deterministic AE space. The generated properties closely follow the requested targets for both logP and molecular weight, showing that the representation originally learned for unconditional graph generation already provides a suitable continuous domain for property steering. This supports the main premise of our framework: controlled generation does not require learning a separate property-specific encoder-decoder.

The variational representation nevertheless improves target fidelity, as summarized in Table 7 in the main text. The MAE decreases from 0.247 to 0.154 for logP and from 3.923 to 2.578 for MW, together with increased Pearson correlations. These results suggest that the VAE latent space is more amenable to continuous property steering. This observation is also consistent with the molecular property-optimization experiments reported for GraViti, which found the variational latent representation to provide a favorable domain for navigating molecular properties.

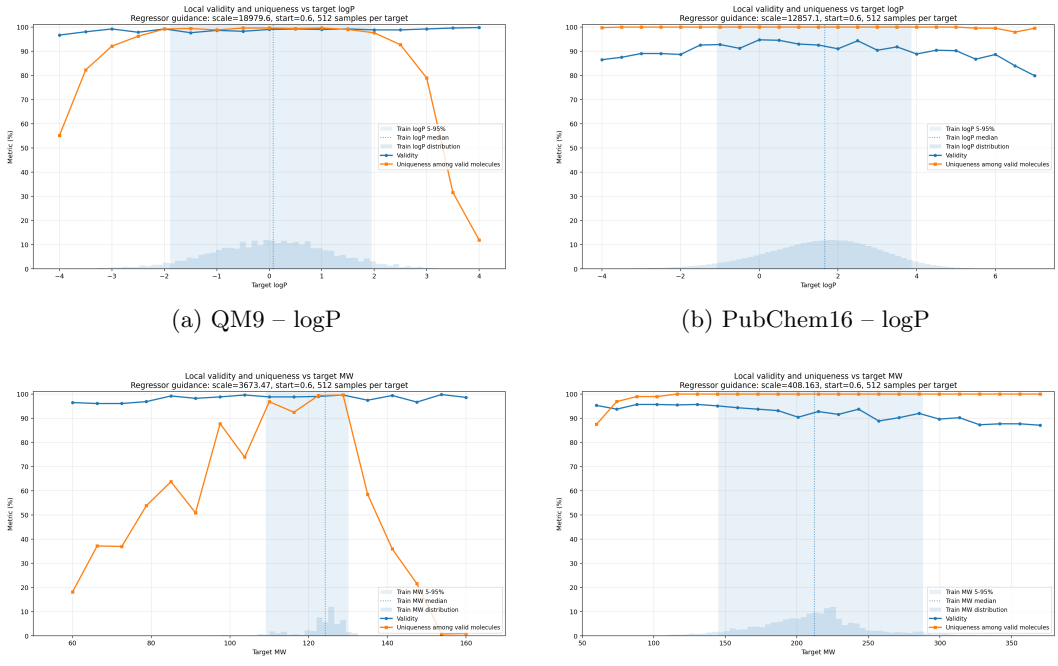


(a) logP (b) Molecular weight

Figure 3: Independent property-controlled generation in the deterministic GraViti AE latent space on PubChem16. Generated properties are shown as a function of the requested target for logP and molecular weight. The dashed line denotes ideal control ($y = x$), and the shaded region indicates the corresponding training-distribution range.

B Validity under Property Control

Figure 4 shows how validity and uniqueness evolve with the requested property. Both remain high within the training distribution, confirming that property control does not substantially degrade generation quality over the range observed during training. Validity decreases mainly for rarer or more extreme targets, revealing where stronger steering begins to move samples toward latent regions that decode less reliably. These results also qualitatively suggest that the models retain some robustness beyond the training distribution.



(a) QM9 – logP

(b) PubChem16 – logP

(c) QM9 – Molecular Weight

(d) PubChem16 – Molecular Weight

Figure 4: Validity and uniqueness as a function of the target property for property-guided molecular generation on QM9 and PubChem16.

C Extended CFM Training on PubChem16

To assess whether the performance of Categorical Flow Maps on PubChem16 is primarily limited by training duration, we additionally train the CSD and ECLD variants for 30 epochs, corresponding to approximately 107 GPU hours. Table 5 reports the resulting unconditional generation performance across different numbers of sampling steps.

Table 5: Unconditional generation results for CFM-CSD and CFM-ECLD on PubChem16 after 30 training epochs (∼107 GPU hours). Validity, uniqueness, and novelty are reported as mean ± standard deviation.

Method	**Steps**	**Validity ↑**	**Uniqueness ↑**	**Novelty ↑**	**FCD ↓**
CFM-CSD	1	88.19 ± 1.11	99.51 ± 0.23	75.10 ± 1.30	6.8832
CFM-CSD	2	86.48 ± 0.97	99.98 ± 0.07	87.03 ± 1.01	2.5352
CFM-CSD	5	93.85 ± 0.75	100.00 ± 0.00	88.58 ± 0.93	1.4919
CFM-CSD	20	97.58 ± 0.51	99.99 ± 0.02	88.77 ± 1.00	0.9430
CFM-ECLD	1	96.02 ± 0.55	98.87 ± 0.35	64.62 ± 1.47	6.8654
CFM-ECLD	2	94.45 ± 0.64	99.96 ± 0.06	85.91 ± 1.27	2.6004
CFM-ECLD	5	96.75 ± 0.62	99.99 ± 0.02	88.20 ± 0.78	1.8569
CFM-ECLD	20	97.63 ± 0.40	99.98 ± 0.04	88.06 ± 0.93	1.6251

D Global Validity under Property Control

Global validity evaluates property-guided generation when target properties are sampled according to their empirical training distribution. Under this setting, property guidance preserves high validity and diversity on both datasets, as reported in Table 6. On QM9, validity remains close to the unconditional regime for both properties, while uniqueness also remains high. On PubChem16, validity stays above 0.92 for both logP and MW and uniqueness remains maximal.

Table 6: Global validity and uniqueness under property-guided generation. Target properties are sampled according to their empirical training distribution before generating the corresponding molecules.

Dataset	**Property**	**Validity ↑**	**Uniqueness ↑**
QM9	logP	0.9873	0.996
QM9	MW	0.9824	0.998
PubChem16	logP	0.9277	1.000
PubChem16	MW	0.9268	1.000

E Flow Matching

E.1 Basic principle

Flow Matching (Lipman et al., 2022) learns a continuous transport between a simple source distribution p_0 and a target distribution p_1. In our setting, p_1 is the distribution of encoded graph representations. Given $z_0 \sim p_0$, $z_1 \sim p_1$, and $t \sim \mathcal{U}[0,1]$, we consider the linear probability path

$$z_t = (1-t)z_0 + tz_1, \tag{8}$$

with conditional velocity

$$u_t(z_t \mid z_0, z_1) = z_1 - z_0. \tag{9}$$

A neural vector field $v_\theta(z_t, t)$ is trained through the conditional flow matching objective

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,z_0,z_1}\left[\|v_\theta(z_t,t) - (z_1 - z_0)\|_2^2\right]. \tag{10}$$

At inference time, samples are generated by integrating the learned vector field from $t = 0$ to $t = 1$ and decoding the resulting latent representations. We additionally consider mini-batch optimal-transport couplings between source and target samples, following previous flow matching work, to construct more coherent training paths.

E.2 Guided generation

A continuous latent representation also provides a direct interface for controlled generation. Conditional flow matching augments the vector field with a target property c,

$$v_\theta(z_t, t, c), \tag{11}$$

while post-hoc guidance can further modify the sampling dynamics using gradients from a differentiable predictor. Given a predictor $r_\omega(z_t, t)$ and a target c, the guided field takes the form

$$\tilde{v}_\theta(z_t,t,c) = v_\theta(z_t,t,c) - \lambda \nabla_{z_t}\mathcal{L}_{\mathrm{prop}}\left(r_\omega(z_t,t), c\right), \tag{12}$$

where λ controls the guidance strength. This follows the general principle of gradient-based generative guidance (Dhariwal & Nichol, 2021) and enables graph-level properties to be controlled directly through the continuous latent dynamics.

E.3 ODE Solvers

All latent flow models are sampled by numerically integrating the learned ordinary differential equation

$$\frac{dz_t}{dt} = v(z_t, t), \qquad t \in [0,1], \tag{13}$$

where v denotes the learned velocity field, possibly augmented with a guidance term. We consider three explicit integration schemes: Euler, midpoint, and Heun. Let z_n denote the current latent state at time t_n and let $h = t_{n+1} - t_n$.

Euler. The Euler method performs a single evaluation of the velocity field per integration step:

$$z_{n+1} = z_n + h\, v(z_n, t_n). \tag{14}$$

Midpoint. The midpoint method is a second-order Runge–Kutta scheme. It first evaluates the velocity at the beginning of the step,

$$k_1 = v(z_n, t_n), \tag{15}$$

and uses it to estimate the state at the midpoint,

$$z_{n+\frac{1}{2}} = z_n + \frac{h}{2}k_1. \tag{16}$$

The velocity is then evaluated at this intermediate state,

$$k_2 = v\left(z_{n+\frac{1}{2}}, t_n + \frac{h}{2}\right), \tag{17}$$

and the update is given by

$$z_{n+1} = z_n + h\, k_2. \tag{18}$$

Heun. Heun's method is also a second-order Runge–Kutta scheme. It first computes an Euler prediction,

$$k_1 = v(z_n, t_n),\ \tilde{z}_{n+1} = z_n + h\,k_1, \tag{19}$$

then evaluates the velocity at the predicted endpoint,

$$k_2 = v(\tilde{z}_{n+1}, t_{n+1}), \tag{20}$$

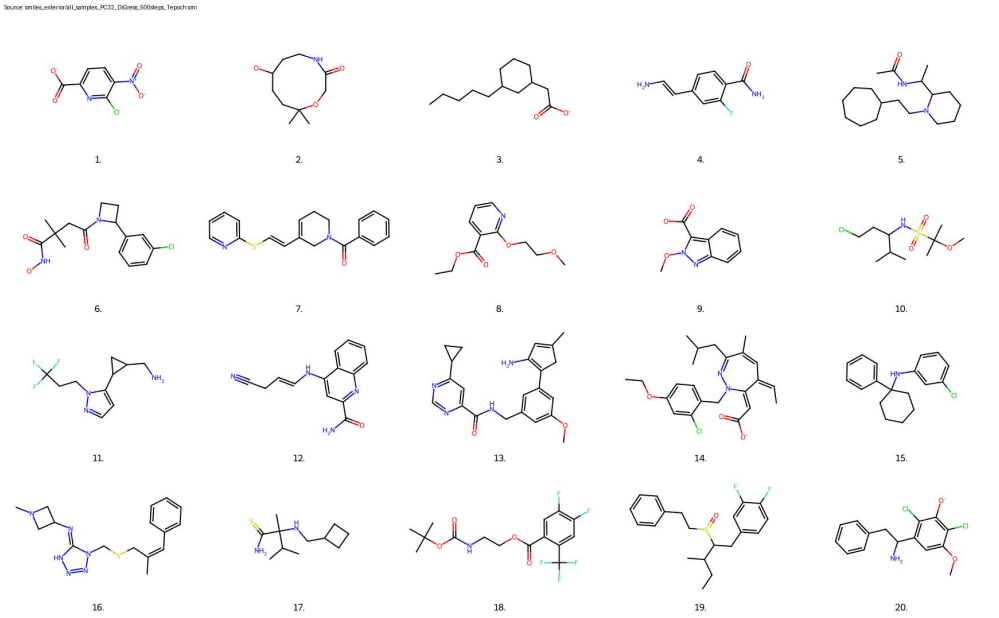
and averages the two slopes:

$$z_{n+1} = z_n + \frac{h}{2}(k_1 + k_2). \tag{21}$$

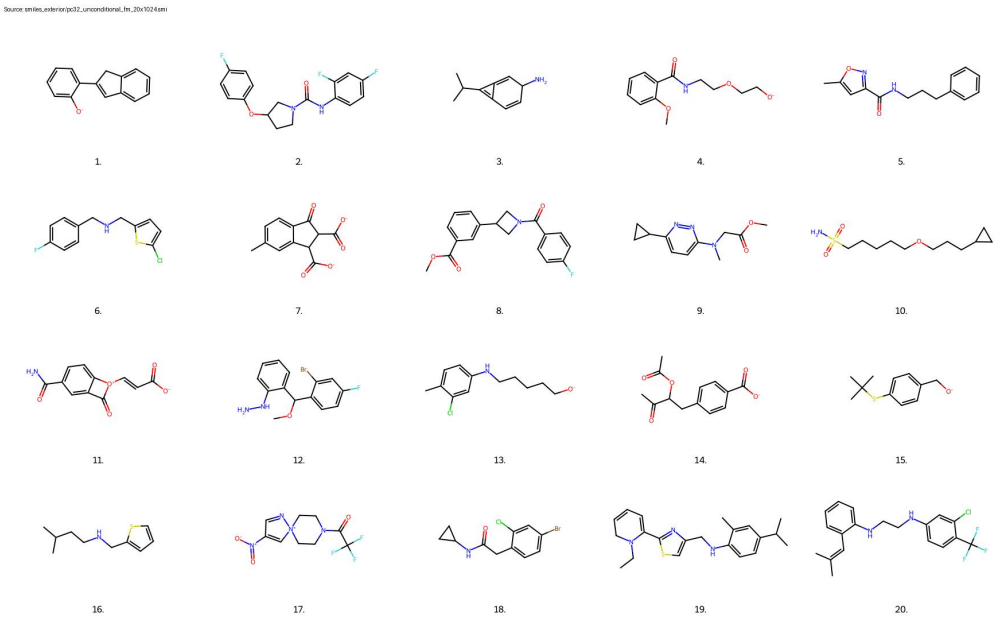
Number of function evaluations. Euler requires one velocity-field evaluation per integration step, whereas midpoint and Heun require two. Consequently, S Euler steps correspond to S NFE, while S midpoint or Heun steps correspond to $2S$ NFE.

F Qualitative examples on PubChem32

Figure 5 shows 20 example molecules generated on PubChem32 using DiGress and our latent Flow Matching model.



(a) 20 molecules generated by DiGress with 500 sampling steps.



(b) 20 molecules generated by our Flow Matching model with 20 sampling steps.

Figure 5: Qualitative examples of molecules generated on PubChem32 by DiGress (top) and our Flow Matching model (bottom).

G Examples of Property-Guided Generation

Figure 6 provides qualitative examples of molecules generated under property guidance for molecular weight (MW) and logP. For targets within the training distribution, generated molecules generally remain close to the requested property value. As targets move outside the training distribution, larger deviations from the target can be observed, illustrating the increased difficulty of extrapolating property control beyond the training regime.

H Influence of the Latent Representation on Property Control

We further investigate whether the choice of latent representation affects property controllability. On PubChem16, we apply the same property-guidance protocol to the deterministic and variational GraViti latent spaces. This comparison isolates the effect of the representation itself, as both spaces are reused directly from the unconditional generation setting, without training a property-specific encoder or decoder.

As shown in Table 7, both representations support effective control of logP and molecular weight (MW). However, the variational representation consistently provides more accurate target matching, achieving lower MAE and higher Pearson correlation for both properties.

Table 7: Effect of the graph-level latent representation on independent property control on PubChem16. Both representations are evaluated using the same guidance protocol.

Representation	**Property**	**MAE ↓**	**r ↑**
AE	logP	0.247	0.966
VAE	logP	**0.154**	**0.978**
AE	MW	3.923	0.971
VAE	MW	**2.578**	**0.977**

These results suggest that the variational representation is more amenable to continuous property steering. This observation is consistent with the property-optimization results reported for GraViti, where the variational latent representation similarly provides a favorable space for navigating molecular properties.

I Training settings

Table 8 summarizes the main architecture, training, and sampling hyperparameters used for the models retained in our experiments.

Table 8: Hyperparameters used for the models retained in our experiments.

Hyperparameter	**QM9NoHydro**	**PubChem16**
Latent dimension	64	256
Latent transform (FM)	None	Standardization
Flow Matching		
Hidden dimension	1024	2048
Number of layers	6	10
Time embedding dimension	1	64
FiLM	Yes	Yes
Batch OT	Yes	Yes
Dropout	0	0
Optimizer	AdamW	AdamW
Learning rate	2×10^{-4}	2×10^{-4}
Weight decay	10^{-5}	10^{-5}
Batch size	1024	1024
Training epochs	500	30
Warmup steps	1000	1000
Gradient clipping	1	1
Validity predictor		
Hidden dimension	256	1024
Number of layers	3	6
Time conditioning	Concatenated to input	Concatenated to input
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-5}	10^{-5}
Batch size	1024	1024
Maximum epochs	300	300
Selected epoch	266	50
Early-stopping patience	50	30
Validity Flow Matching		
Latent transform	Standardization	Standardization
Hidden dimension	1536	2048
Number of layers	8	10
Dropout	0	0
Log-variance bounds	$[-6, 2]$	$[-6, 2]$
Learning rate	3×10^{-4}	3×10^{-4}
Minimum learning rate	10^{-5}	10^{-5}
Weight decay	10^{-5}	10^{-5}
Batch size	512	1024
Maximum epochs	500	150
Selected epoch	500	32
Early-stopping patience	500	5
Endpoint candidates K	8	32
Reweighting strength β	1	5
Sampling		
Integrator	Euler	Euler
Integration interval	$[0, 1]$	$[0, 1]$
Default FM steps	500	20
VFM steps	20 / 500	20 / 500
Guidance scale	25	10
Guidance start	$t = 0.6$	$t = 0.6$

Table 9: Hyperparameters used for property-guided generation on QM9NoHydro. Conditioning properties are standardized using training-set statistics. No latent transformation is applied.

Hyperparameter	**QM9 MW**	**QM9 logP**
Latent dimension	64	64
Conditioning property	MW	logP
Property normalization	Standardized	Standardized
Latent transform	None	None
Conditional Flow Matching		
Hidden dimension	512	512
Number of residual blocks	6	6
Conditioning mechanism	FiLM	FiLM
Condition dimension	1	1
Dropout	0	0
Batch OT	Yes	Yes
Optimizer	AdamW	AdamW
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-5}	10^{-5}
Batch size	512	512
Training epochs	500	500
Validation split	0.1	0.1
Gradient clipping	0.5	0.5
Property regressor		
Hidden dimension	256	256
Number of hidden layers	3	3
Activation	SiLU	SiLU
Normalization	LayerNorm	LayerNorm
Time conditioning	Concatenated to input	Concatenated to input
Output dimension	1	1
Optimizer	Adam	Adam
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-5}	10^{-5}
Training epochs	300	300
Gradient clipping	1.0	1.0
Batch optimal transport		
Sinkhorn batch size	512	512
Regularization	1.0	1.0
Maximum iterations	1000	1000
Convergence threshold	10^{-6}	10^{-6}
Sampling and guidance		
ODE solver	Midpoint	Midpoint
Number of steps	20	20
Temperature	1.0	1.0
Guidance start	$t = 0.6$	$t = 0.6$
Guidance scale	3673.47	18979.6
Random seed	42	42
Samples per target	512	512
Global evaluation samples	1024	1024

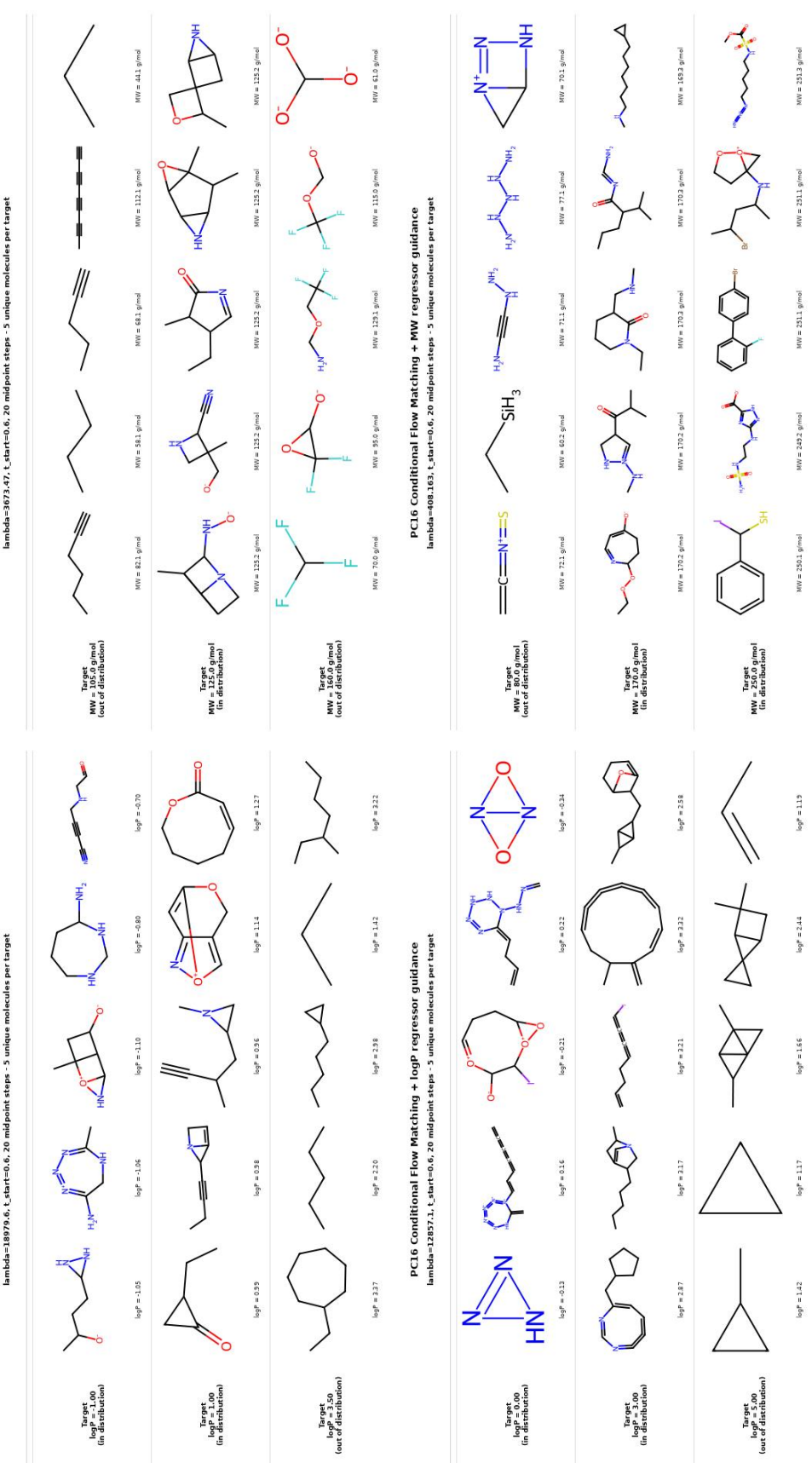


Figure 6: Examples of molecules generated with property-guided latent Flow Matching on QM9 and PubChem16 for logP and molecular weight (MW) targets. Examples are shown for both in-distribution and out-of-distribution target values. The achieved property value is reported below each generated molecule.

Table 10: Hyperparameters used for property-guided generation on PubChem16. Conditioning properties are standardized using training-set statistics. No latent transformation is applied.

Hyperparameter	**PubChem16 MW**	**PubChem16 logP**
Latent dimension	256	256
Conditioning property	MW	logP
Property normalization	Standardized	Standardized
Latent transform	None	None
Conditional Flow Matching		
Hidden dimension	1024	1024
Number of residual blocks	6	6
Conditioning mechanism	FiLM	FiLM
Condition dimension	1	1
Dropout	0	0
Batch OT	Yes	Yes
Optimizer	AdamW	AdamW
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-5}	10^{-5}
Batch size	512	512
Training epochs	1000	1000
Validation split	0.1	0.1
Gradient clipping	0.5	0.5
Property regressor		
Hidden dimension	256	256
Number of hidden layers	3	3
Activation	SiLU	SiLU
Normalization	LayerNorm	LayerNorm
Time conditioning	Concatenated to input	Concatenated to input
Output dimension	1	1
Optimizer	Adam	Adam
Learning rate	3×10^{-4}	3×10^{-4}
Weight decay	10^{-5}	10^{-5}
Training epochs	300	300
Gradient clipping	1.0	1.0
Batch optimal transport		
Sinkhorn batch size	512	512
Regularization	1.0	1.0
Maximum iterations	1000	1000
Convergence threshold	10^{-6}	10^{-6}
Sampling and guidance		
ODE solver	Midpoint	Midpoint
Number of steps	20	20
Temperature	1.0	1.0
Guidance start	$t = 0.6$	$t = 0.6$
Guidance scale	408.163	12857.1
Random seed	42	42
Samples per target	512	512
Global evaluation samples	1024	1024