One-Step Generative Modeling with Mean Flows for Molecular Structures

Silpa Soni Nallacheruvu ssnal@kth.se
Tove Nordmark toveno@kth.se
Lena Weyer weyer@kth.se

Abstract

We investigate one-step generative modeling with Mean Flows under data-constrained regimes and extend the framework to structured molecular latent spaces. Mean Flows learn average velocity fields between time pairs, enabling single-step (1-NFE) sampling. While prior work demonstrated strong performance at large scale, its behavior under limited data and reduced training budgets remains unexplored. We conduct systematic ablation studies on a 5,000-image ImageNet subset, analyzing time-pair sampling, Jacobian–vector product (JVP) formulation, positional embeddings, adaptive loss weighting, and classifier-free guidance. We observe scale-dependent deviations from previously reported optima, revealing that certain design choices (e.g., full $r \neq t$ sampling) become preferable in low-data regimes.

Beyond images, we adapt Mean Flows to the 56-dimensional latent space of a Junction Tree VAE for molecular generation. PCA-based distribution analysis shows that the learned flow reshapes Gaussian priors toward the encoded molecular manifold. One-step sampling yields chemically valid molecules (99.6% validity), demonstrating that Mean Flows can operate on structured scientific latent spaces. Our results suggest that one-step generative flows remain stable under constrained training and can generalize beyond image domains.

1 Introduction

Recent generative models based on diffusion and flow-matching frameworks achieve high sample quality but typically require many sequential neural network evaluations at inference time. This multi-step sampling procedure increases computational cost and limits applicability in settings where fast generation is essential. Mean Flows were proposed as a one-step alternative that predicts average velocity fields between two time points rather than instantaneous velocities, enabling single-function-evaluation (1-NFE) sampling while preserving competitive generation quality.

While the original Mean Flows work demonstrates strong large-scale performance, its behavior under constrained data and limited compute has not been systematically examined. Understanding how its components behave outside the high-resource regime is important for evaluating robustness and identifying which design choices are fundamentally necessary versus scale-dependent. In this work, we reimplement the Mean Flows training pipeline and conduct controlled ablation studies on a reduced ImageNet subset. We analyze sensitivity to time-pair sampling, Jacobian–vector product (JVP) formulation, positional embeddings, adaptive loss weighting, classifier-free guidance, and model scaling. Our results show that several hyperparameter optima shift under data constraints, providing insight into how Mean Flows behave away from the large-scale regime.

We further investigate whether the Mean Flows formulation can extend beyond image generation. Specifically, we adapt the framework to operate in the 56-dimensional latent space of a Junction Tree VAE for molecular graph generation. Through latent space visualization and validity analysis, we evaluate whether a one-step flow can meaningfully reshape Gaussian noise toward structured molecular manifolds.

HT20 Deep Learning Advanced, DD2412, KTH.

By combining systematic reproduction with a domain transfer experiment, this work aims to clarify the robustness, sensitivity, and portability of one-step generative flows.

2 Related Work

Modern generative modeling has been shaped by diffusion models and continuous-time flow-based approaches. Flow Matching provides a unified framework for learning time-dependent vector fields, enabling stable training and high-quality generation. Mean Flows [1] extend this idea by predicting average rather than instantaneous velocities through a two-time conditioning formulation and a Jacobian–vector-product objective, allowing high-fidelity image synthesis in a single function evaluation. The original paper conducts extensive ablations on ImageNet, studying the effects of time-pair sampling, positional embeddings, loss metrics, and classifier-free guidance. We conducted a controlled re-evaluation of Mean Flows under low-data and constrained-compute regimes, which were not analyzed in the original work.

The Diffusion Transformer (DiT) architecture has become a strong backbone for diffusion and flow-based models. We follow the open-source implementation of Peebles and Xie [2] to construct a JAX/Flax DiT compatible with Mean Flows, including patch embeddings, sinusoidal position encodings, and adaLN-Zero conditioning. Image generation quality is evaluated using Fréchet Inception Distance (FID), computed from Inception-V3 features [3], comparing generated samples to the full ImageNet distribution.

Beyond images, generative modeling of molecular structures is an expanding research frontier. Models such as Proteina [4] demonstrate how flow-based methods can learn complex structural manifolds in biomolecular systems. Graph-based molecular generation frequently relies on structured latent spaces, and the Junction Tree Variational Autoencoder (JT-VAE) [5] is among the most widely adopted methods for ensuring chemical validity. We use the open-source JT-VAE implementation[1] [6] to obtain latent representations suitable for flow-based generation.

Building on these foundations, our project explores whether the Mean Flows formulation, originally proposed for large-scale image synthesis can be adapted to molecular latent spaces. By integrating DiT and JT-VAE within the Mean Flow framework, we provide an initial demonstration of one-step molecular generation, connecting image-based flow research to emerging scientific applications.

3 Data

For our replication experiments, we used a reduced version of ImageNet (ILSVRC2012), selecting 5,000 images uniformly across 1,000 classes to fit computational limits. Images were resized, normalized, and converted into latent representations using the pretrained StabilityAI VAE tokenizer (`sd-vae-ft-mse`), consistent with the Mean Flows setup.

For molecular generation, we used the QM9 and ZINC datasets, which provide small organic molecules in SMILES format with atom-level information. These data required conversion into the junction-tree representation needed by JT-VAE. The molecular datasets were not used to train Mean Flows directly; instead, they supplied latent vectors to the JT-VAE pipeline, which we integrated with a Mean-Flows style sampler for molecule generation.

4 Methods

4.1 Mean Flows for One-Step Generation

Mean Flows build upon the Flow Matching framework by replacing instantaneous velocity prediction with the prediction of *average velocity fields* between two time points. Let z_t denote a point along a continuous trajectory governed by an unknown velocity field $v(z_t, t)$. Instead of directly learning $v(z_t, t)$, Mean Flows define the average velocity between times r and t as

$$u(z_t, r, t) = \frac{1}{t-r} \int_r^t v(z_\tau, \tau)\, d\tau. \tag{1}$$

This formulation enables a one-step generative update by learning a mapping from noise directly to data in a single model evaluation.

[1] https://github.com/VldKnd/vae_qm9

Training relies on the identity

$$u(z_t, r, t) = v(z_t, t) - (t - r)\,\frac{d}{dt}u(z_t, r, t), \tag{2}$$

which expresses the average velocity in terms of the instantaneous velocity and its time derivative. The time derivative is computed efficiently using a Jacobian–vector product (JVP) without explicitly forming the Jacobian:

$$\frac{d}{dt}u(z_t, r, t) = \frac{dz_t}{dt}\partial_z u + \frac{dr}{dt}\partial_r u + \frac{dt}{dt}\partial_t u. \tag{3}$$

Under the training dynamics, we have $dz_t/dt = v(z_t, t)$, $dr/dt = 0$, and $dt/dt = 1$, yielding

$$\frac{d}{dt}u(z_t, r, t) = v(z_t, t)\partial_z u + \partial_t u. \tag{4}$$

The model u_θ is trained to minimize the mean squared error

$$\mathcal{L} = \mathbb{E}_{(r,t)}\left\|u_\theta(z_t, r, t) - \tilde{u}(z_t, r, t)\right\|_2^2, \tag{5}$$

where $\tilde{u}$ is computed from the identity above. Sampling then consists of a single update step using the learned average velocity field.

This formulation allows controlled ablations over:

- the sampling distribution of time pairs (r, t),
- the correctness of the JVP tangent,
- the representation of temporal embeddings,
- adaptive loss weighting schemes, and
- classifier-free guidance strength.

4.2 DiT Backbone for Velocity Parameterization

To parameterize $u_\theta(z_t, r, t)$, we implement a Diffusion Transformer (DiT) backbone operating in the latent space of a pretrained VAE tokenizer. Input latents are divided into patches and embedded into a token sequence. Fixed sinusoidal positional encodings are added to preserve spatial structure.

Temporal conditioning is incorporated through learned embeddings of t and r, which are injected into transformer blocks via adaptive Layer Normalization (adaLN-Zero). When class-conditional generation is used, label embeddings are incorporated in the same manner.

Each transformer block consists of multi-head self-attention followed by a feed-forward network. After processing, tokens are projected back to latent space and unpatchified to yield the predicted average velocity $u_\theta(z_t, r, t)$.

To enable Mean Flow training, the forward pass is modified to support JVP computation inside the training loop. This requires computing directional derivatives with respect to time while preserving gradient flow for parameter updates.

4.3 Adaptive Loss Weighting

We explore an adaptive loss weighting scheme of the form

$$w = \frac{1}{(\|\Delta\|_2^2 + c)^p}, \tag{6}$$

where $\Delta = u_\theta - \tilde{u}$, c is a small constant for numerical stability, and p controls the strength of reweighting.

Unlike the original formulation, we normalize the weights within each batch:

$$\tilde{w}_i = \frac{w_i}{\sum_j w_j}. \tag{7}$$

This preserves relative weighting between samples while preventing large variations in global loss scale across different values of p. This modification changes the optimization objective up to a multiplicative normalization factor and therefore produces results that are not numerically comparable to the original Mean Flows study. In Table 1e, we analyze how this normalization affects the behavior observed in ablation studies.

4.4 Molecular Latent Space Modeling with JT-VAE

To investigate domain transfer, we extend Mean Flows to molecular generation. We first train a Junction Tree Variational Autoencoder (JT-VAE) to encode molecular graphs into a structured 56-dimensional latent representation composed of tree and graph components.

The JT-VAE is trained using a two-stage protocol:

1. A pretraining phase without KL regularization ($\beta = 0$) to stabilize reconstruction.
2. A variational phase with KL regularization ($\beta = 0.005$).

After training, molecules are encoded into latent vectors. The generative task is thus reformulated as modeling the latent distribution rather than molecular graphs directly.

4.5 Mean Flows in Molecular Latent Space

We replace the image-based DiT backbone with a molecule-specific transformer operating directly on the 56-dimensional JT-VAE latent vectors.

Each latent vector is embedded into a small token sequence and augmented with 1D sinusoidal positional encodings. Unlike the image setting, no class conditioning is used; the model is conditioned only on temporal variables (r, t).

The molecular DiT predicts an updated latent vector of identical dimensionality. Training follows the same Mean Flow objective described in Section 4.1.

Sampling proceeds as follows:

1. Draw $z_0 \sim \mathcal{N}(0, I)$.
2. Apply one Mean Flow update to obtain z_1.
3. Decode z_1 using the pretrained JT-VAE decoder to obtain a molecular graph.

4.6 Latent Space Distribution Analysis

To evaluate whether the learned flow reshapes the prior distribution toward the molecular data manifold, we perform principal component analysis (PCA) on the 56-dimensional JT-VAE latents.

PCA is fitted on encoded dataset latents, defining a shared two-dimensional projection space. We then project:

- encoded dataset latents,
- Gaussian prior samples,
- Mean Flow–generated latents.

This enables visual comparison of distribution overlap and structural alignment. We additionally compare marginal distributions along principal components and compute nearest-neighbor distances in PCA space to quantify how the learned flow transforms noise toward the molecular data manifold.

5 Experiments and findings

5.1 Ablation Studies

Using the hyper parameter settings listed in subsection 8.1, we ran ablation studies to see how different parts of the model influence one-step performance as is shown in Table 1.

From Flow Matching to Mean Flows:

Table 1a shows how performance varies with the proportion of $r \neq t$ samples. The 0% case reduces to standard Flow Matching, which performs worst (FID 131.94), confirming that instantaneous velocities alone are insufficient for effective one-step generation. FID improves monotonically as the $r \neq t$ fraction increases, with 100% achieving the best result (FID 114.48). This suggests that under our training regime, the model benefits more from learning average velocities across diverse time intervals than from mixing in instantaneous velocity information, which may introduce noise without sufficient data to learn precise estimates.

Table 1: Ablation study on 1-NFE ImageNet 256×256 generation. **FID-1K** is reported. Default configs are light gray. Best FID in bold.

(a) Ratio $r \neq t$

% $r \neq t$	FID
0% (=FM)	131.94
25%	126.31
50%	121.98
100%	**114.48**

(b) JVP computation

Tangent	FID
$(v, 0, 1)$	**126.31**
$(v, 0, 0)$	138.45
$(v, 1, 0)$	138.15
$(v, 1, 1)$	139.02

(c) Positional embedding

Embedding	FID
(t, r)	127.35
$(t, t-r)$	126.31
$(t, r, t-r)$	**125.86**
$t-r$ only	131.08

(d) Time samplers

Sampler	FID
uniform(0,1)	126.76
lognorm$(-0.2, 1.0)$	122.03
lognorm$(-0.2, 1.2)$	127.78
lognorm$(-0.4, 1.0)$	126.31
lognorm$(-0.4, 1.2)$	**119.56**

(e) Loss metrics

p	FID
0.0	128.99
0.5	**124.90**
1.0	126.31
1.5	130.47
2.0	128.25

(f) CFG scale

ω	FID
1.0 (no cfg)	126.31
1.5	**118.39**
2.0	129.94
3.0	128.69
5.0	129.33

JVP Computation:

Table 1b demonstrates the critical importance of correct Jacobian–vector product computation. The proper tangent $(v, 0, 1)$ corresponds to the time derivative decomposition $(\partial_z u,\ \partial_r u,\ \partial_t u)$ from subsection 4.1. All three incorrect variants produce significantly worse FID, with performance degrading by approximately 10%. This confirms that even small deviations from the theoretically correct JVP formulation corrupt the velocity field estimates and severely impact generation quality.

Conditioning on (r, t):

Table 1c shows that the model is robust to different positional embedding schemes, with all variants achieving reasonable performance. Using all three terms $(t, r, t-r)$ performs marginally best, followed closely by $(t, t-r)$. Embedding only the interval $t-r$ performs worst, suggesting that absolute time information is valuable. The small differences indicate that as long as both time variables are encoded, the specific representation matters less than their presence.

Time Samplers:

Table 1d confirms that the time-sampling distribution significantly affects training. Logit-normal samplers consistently outperform uniform sampling, with lognorm$(-0.4, 1.2)$ achieving the best result. These distributions bias sampling toward intermediate and later times, which may provide more informative training signals for learning the average velocity field. The sensitivity to sampler parameters ($\sigma = 1.0$ vs. 1.2) suggests this was a critical hyperparameter in reduced training dataset.

Loss Metrics:

Our normalized adaptive weighting shows clear sensitivity to the power parameter p (Table 1e). Standard MSE ($p = 0$) performs reasonably, but modest adaptive weighting ($p = 0.5$) improves performance, likely by allowing the model to focus more on challenging samples without completely ignoring easy ones. Higher values ($p \geq 1.5$) degrade performance, suggesting excessive downweighting of large errors causes the model to neglect important learning signals. The narrow optimal range around $p = 0.5$–1.0 indicates that balanced weighting is crucial under our training constraints.

Guidance Scale:

Table 1f shows that CFG substantially improves generation quality, with $\omega = 1.5$ achieving the best FID (118.39)—a 6% relative improvement over no guidance (FID 126.31). However, stronger

Figure 2: Comparison between random batches of real ImageNet images (left) and our one-step generated samples (right) with FID-1K = 120.22

guidance ($\omega \geq 2.0$) degrades performance. This suggests our model has learned a useful but noisy distinction between conditional and unconditional predictions. Moderate amplification strengthens class-specific features, but excessive amplification ($\omega \geq 3.0$) magnifies noise, likely because our limited training data (5 images per class) prevents learning sharp class boundaries.

Scalability:

Across model sizes (B/2, M/2, L/2), FID remains stable for 7–8 epochs before diverging sharply (Figure 1), with larger models degrading earlier and more severely. This pattern indicates overfitting: with only 5,000 training images, larger models quickly memorize the dataset rather than learning generalizable flows. Since FID is evaluated against the full ImageNet distribution, this train-test distribution mismatch becomes increasingly penalized. The instability demonstrates that under severe data constraints, we do not observe the monotonic scaling behavior reported at large scale.

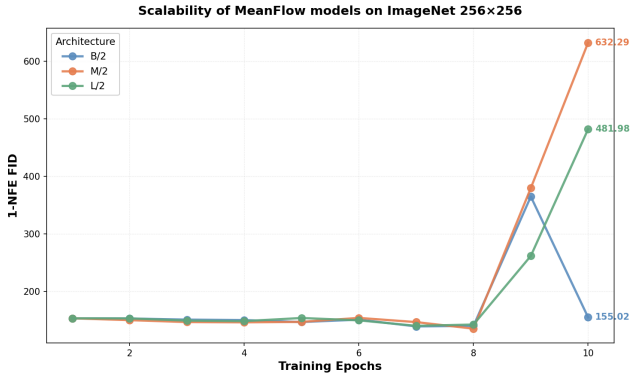

Figure 1: Scalability of Mean Flow models (DiT-B/2, M/2, L/2) on Imagenet 256 × 256

5.2 Imagenet Generation

Using the combined best settings (listed in subsection 8.2) from our ablation studies, we trained a final model for 30 epochs to inspect one-step generated samples. The resulting FID of 120.22, slightly worse than the best isolated FID of 114.48, suggests that these hyperparameters interact in non-additive ways and that individually optimal choices do not necessarily form an optimal joint configuration. As shown in Figure 2, the generated images lack fine details and sharp structure but still capture coarse shapes, colour patterns, and approximate object layouts. This indicates that, even with highly reduced training, the model learns a meaningful one-step mapping from noise toward ImageNet-like images.

5.3 Molecular Structure Generation

The molecular DiT model (see subsection 8.3) was trained for 90 epochs using a 90/10 train–validation split of the JT-VAE encoded latent dataset. Figure 3 presents the training and validation losses. Both curves exhibit an overall decreasing trend, indicating successful optimization

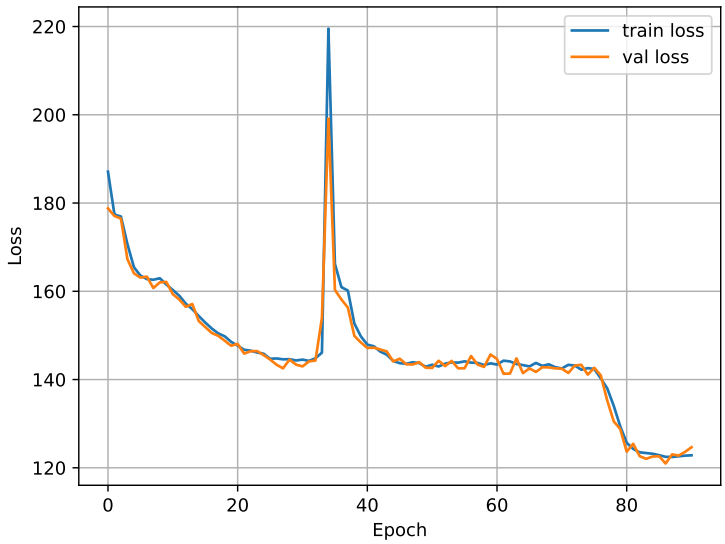

Figure 3: Training and validation loss curves for the molecular Mean Flow model.

Data set	499 / 500 (99.80%)
Prior	496 / 500 (99.20%)
Mean Flow model	498 / 500 (99.60%)

Table 2: Percentage of valid decoded molecules (out of 500 samples) from dataset latents, Gaussian prior samples, and Mean Flow–generated latents.

of the Mean Flow objective in latent space. A transient instability is observed around epoch 35, where both training and validation losses momentarily increase. This behavior may be attributed to optimizer hyperparameters (e.g., learning rate or β_2 in Adam) or sensitivity of the flow objective to latent-space curvature. Nevertheless, training stabilizes afterward and converges to a lower loss regime.

Reconstruction and Validity. The pretrained JT-VAE achieved 22% exact reconstruction accuracy, defined as the proportion of molecules whose decoded structure exactly matches the input molecule. While this metric reflects strict reconstruction fidelity, it does not measure chemical validity.

To assess validity, we sampled 500 latent vectors from three sources: (i) encoded dataset latents, (ii) the standard Gaussian prior $\mathcal{N}(0, I)$, and (iii) the trained Mean Flow model. Each latent was decoded using the JT-VAE decoder and checked for dictionary-valid molecular structures. Results are shown in Table 2.

Despite modest exact reconstruction accuracy (22%), Mean Flow–generated samples achieve 99.6% chemical validity, slightly exceeding the 99.2% validity obtained by sampling directly from the Gaussian prior. This suggests that the learned one-step transformation preserves latent structure compatible with the JT-VAE decoder and does not degrade chemical validity.

Latent Distribution Alignment. To evaluate whether the Mean Flow model reshapes the prior toward the molecular data manifold, we performed principal component analysis (PCA) on the 56-dimensional JT-VAE latent space. PCA was fitted on encoded dataset latents, defining a shared projection space. We then projected (i) encoded dataset latents, (ii) Gaussian prior samples, and (iii) Mean Flow–generated latents into this space.

Figure 4 illustrates the transformation from noise to model samples and finally to the encoded data distribution. The prior occupies a broader, less structured region, while flow-generated latents shift toward the high-density regions occupied by encoded data. This indicates that the learned velocity field guides samples toward the target latent manifold.

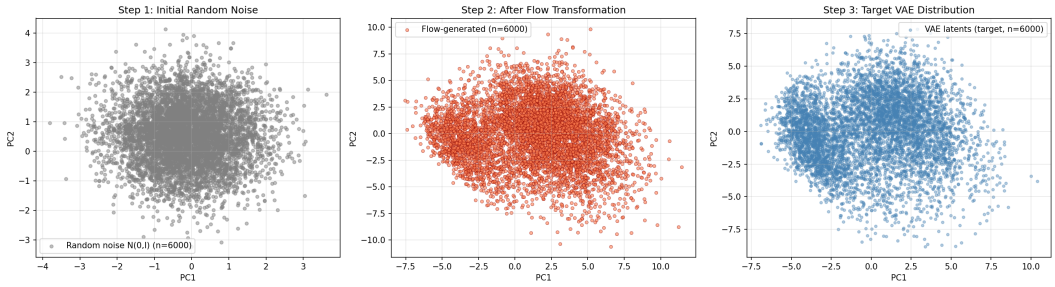

Figure 4: PCA projection of latent distributions (n = 6000). From left to right: Gaussian prior samples, Mean Flow–generated latents, and encoded dataset latents.

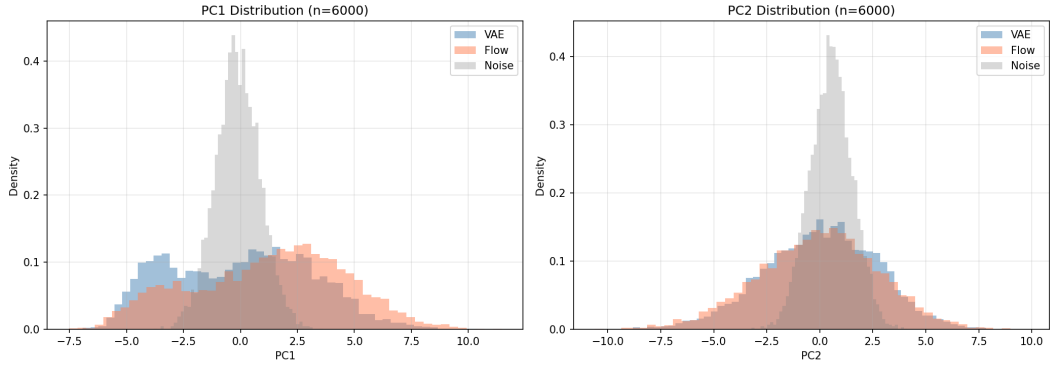

Figure 5: Marginal distributions along the two leading principal components for prior, Mean Flow–generated, and encoded dataset latents.

We further compare marginal densities along the first two principal components in Figure 5. The flow-generated distribution more closely matches the encoded data distribution than the isotropic Gaussian prior, particularly along the dominant principal direction. This supports the conclusion that the Mean Flow transformation reduces distributional discrepancy between noise and the molecular latent manifold.

Qualitative Samples. Representative decoded molecules are shown in Figure 6. Compared to prior samples, Mean Flow–generated molecules exhibit structured ring systems and coherent functional group arrangements similar to dataset examples. Although the model has not been optimized for property control or diversity metrics, the qualitative outputs demonstrate that one-step latent transport can produce chemically coherent structures.

Overall, these results indicate that Mean Flows can successfully operate in structured molecular latent spaces. The learned one-step mapping shifts Gaussian noise toward the JT-VAE data manifold while maintaining high chemical validity. We provide preliminary distributional evidence rather than full molecular benchmarking. Although further evaluation using diversity and property-based metrics is necessary for practical molecular design, this proof-of-concept demonstrates that one-step generative flows extend beyond images to chemically structured domains.

6 Discussion and Conclusion

This work provides a structured reproduction and analysis of the Mean Flows framework under constrained computational settings, together with an exploratory extension to molecular latent spaces. Our ablation studies demonstrate that the core architectural and algorithmic components of Mean Flows are not incidental design choices but essential elements for stable one-step generation. In particular, correct Jacobian–vector product computation, appropriate time-pair sampling, and explicit temporal conditioning significantly influence performance. Even at reduced scale, the qualitative trends observed across ablations are consistent with the theoretical motivations of the original method, reinforcing the internal coherence of the formulation.

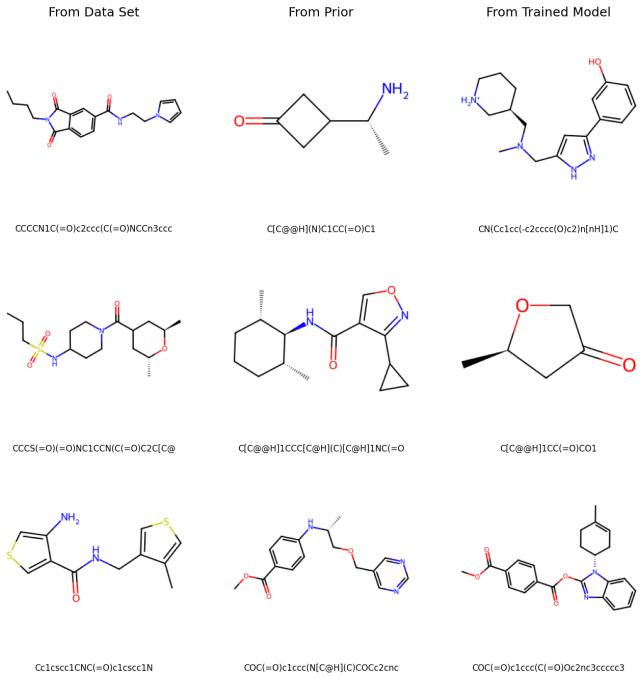

Figure 6: Example molecules sampled from the dataset (left), Gaussian prior (middle), and Mean Flow model (right).

Our results also highlight the sensitivity of one-step generative modeling to hyperparameter interactions. Individually optimal configurations do not necessarily compose into a globally optimal model, suggesting nontrivial coupling between time sampling, guidance strength, and adaptive loss weighting. This underscores the importance of joint hyperparameter tuning in practical deployments of Mean Flows.

Beyond image synthesis, we demonstrate that the Mean Flow objective can be transferred to structured molecular latent spaces. By integrating a transformer-based velocity model with JT-VAE representations, we show that a single learned update can map Gaussian noise toward the molecular data manifold while preserving chemical validity. Although preliminary, these findings indicate that one-step generative flows can operate in scientifically structured domains and may offer computational advantages for large-scale molecular exploration.

Future work should investigate larger molecular datasets, richer chemical validity metrics, property-conditional generation, and extension to protein backbone modeling. More broadly, our findings suggest that continuous-time flow formulations based on averaged dynamics may generalize across modalities when suitable latent representations are available.

7 Limitations

Several limitations should be considered when interpreting our results. First, due to computational constraints, ImageNet experiments were conducted on a 5,000-image subset with substantially reduced training duration relative to the original Mean Flows work. This restricts direct numerical comparability and likely alters convergence behavior, particularly in scalability experiments. Larger models exhibited early overfitting under this regime, preventing full assessment of scaling trends.

Second, we introduced normalized adaptive loss weighting, which modifies optimization dynamics relative to the original formulation. While this improved stability, it affects strict comparability of ablation results.

Third, molecular evaluation relied primarily on reconstruction accuracy, chemical validity, and PCA-based latent alignment. More comprehensive metrics—such as diversity, novelty, distributional similarity (e.g., Fréchet ChemNet Distance), or property-based evaluation—were not included. Addi-

tionally, transient instability observed during molecular training suggests sensitivity to optimizer hyperparameters.

Finally, the molecular extension serves as a proof-of-concept rather than a fully optimized generative pipeline. Further investigation is required to assess scalability, robustness, and applicability to more complex chemical or biological structures.

References

[1] Zhengyang Geng et al. *Mean Flows for One-step Generative Modeling*. 2025. arXiv: 2505.13447 [cs.LG]. URL: https://arxiv.org/abs/2505.13447.

[2] William Peebles and Saining Xie. *Scalable Diffusion Models with Transformers*. 2023. arXiv: 2212.09748 [cs.CV]. URL: https://arxiv.org/abs/2212.09748.

[3] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV]. URL: https://arxiv.org/abs/1512.00567.

[4] Tomas Geffner et al. *Proteina: Scaling Flow-based Protein Structure Generative Models*. 2025. arXiv: 2503.00710 [cs.LG]. URL: https://arxiv.org/abs/2503.00710.

[5] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. *Junction Tree Variational Autoencoder for Molecular Graph Generation*. 2019. arXiv: 1802.04364 [cs.LG]. URL: https://arxiv.org/abs/1802.04364.

[6] Vladimir Kondratyev et al. *Generative model based on junction tree variational autoencoder for HOMO value prediction and molecular optimization*. 2023. URL: https://doi.org/10.1186/s13321-023-00681-4.

8 Appendix

8.1 Hyperparameters for DiT architecture

We have used the same hyper parameters as mentioned in the Appendix of [1] as shown in Table 3. The training computation details are: NVIDIA L4 GPU, g2-standard-16 (16 vCPUs, 64 GB Memory) and a CPU, n2-standard-32 (32 vCPUs, 128 GB Memory).

Hyperparameter	**Value**
Architecture	DiT-B-4
Epochs	10
Learning rate	1×10^{-4}
Adam β_1	0.9
Adam β_2	0.95
EMA decay	0.9999
p	1.0
CFG scale ω	1.0
Ratio $(r \neq t)$	0.25
JVP tangent	$(v, 0, 1)$
(t, r) embedding	(t, r)
Time sampler	Logit-normal $(-0.4, 1.0)$
seed	42

Table 3: Hyperparameters used for DiT-B-4 training.

8.2 Hyperparameters for the best model for 1-NFE generated samples

We trained a model using the best-performing configuration from each ablation study as shown in Table 4. This combined configuration achieved FID 120.22 after 30 epochs, compared to optimal FID 114.48 obtained during the ratio ablation with default settings for other hyperparameters. This suggests that, for instance, stronger CFG ($\omega = 1.5$) may require different loss weighting than our optimal $p = 0.5$, or the aggressive 100% $r \neq t$ sampling may benefit from different time distributions. Additionally, the longer training (30 vs. 10 epochs) may have induced overfitting, as observed in Figure 1.

Hyperparameter	Value
Architecture	DiT-B-4
Epochs	30
Learning rate	1×10^{-4}
Adam β_1	0.9
Adam β_2	0.95
EMA decay	0.9999
p	0.5
CFG scale ω	1.5
Ratio $(r \neq t)$	1.00
JVP tangent	$(v, 0, 1)$
(t, r) embedding	$(t, r, t - r)$
Time sampler	Logit-normal $(-0.4, 1.2)$
seed	3456

Table 4: Hyperparameters used for the best model DiT-B-4 training.

8.3 Hyperparameters for Molecular DiT training

The architecture "Mol-DiT-B" of the generative model used for molecular structures was a transformer of depth 8 with dimensionality of the hidden layers equal to 512, and with 8 heads. The hyperparameters for training are shown in table 5. These were chosen to be similar to the default hyperparameters used in [1]. The batch size was 8192, chosen based on available virtual memory.

Hyperparameter	Value
Architecture	"Mol-DiT-B"
Epochs	90
lr	$5 \cdot 10^{-}5$
Adam β_1	0.9
Adam β_2	0.99
EMA decay	0.9999
p	0.0
Ratio $(t \neq r)$	0.25
JVP Tangent	$(v, 0, 1)$
(t, r) Embedding	$(t, t - r)$
Time sampler	lognorm$(-0.4, 1.0)$

Table 5: Hyperparameters used for Mol-DiT-B-4 training.

8.4 JT-VAE training

8.4.1 Hyperparameters for JT-VAE training

For molecular generation, the raw dataset could not be processed directly on GPU, requiring all JT-VAE preprocessing to be performed on a dedicated CPU machine (see subsubsection 8.4.2).

Hyperparameter	Value
Hidden size	450
Latent size	56
Message passing depth	3
Batch size	64
Optimizer	Adam
Initial learning rate	10^{-3}
LR scheduler	Exponential decay ($\gamma = 0.9$)
Stereo modeling	Enabled

Table 6: Hyperparameters used for JT-VAE training.

8.4.2 Training computation details

Training of the Junction Tree Variational Autoencoder (JT-VAE) was conducted on a GPU machine (Google Cloud `g2-standard-4`, 4 vCPUs, 16 GB RAM, equipped with one NVIDIA L4 GPU). The training procedure followed the two-stage protocol introduced by Jin, Barzilay, and Jaakkola.

All preprocessing steps were performed on a separate CPU machine (Google Cloud `e2-highmem-8`, 8 vCPUs, 64 GB RAM, x86/64 architecture).