PROPERTY-GUIDED MOLECULAR GENERATION AND OPTIMIZATION VIA LATENT FLOWS

Alexander Arjun Lobo*, **Urvi Awasthi***, **& Leonid Zhukov**
BCG X AI Science Institute
{lobo.alexander, awasthi.urvi, zhukov.leonid}@bcg.com

ABSTRACT

Molecular discovery is increasingly framed as an inverse design problem: identifying molecular structures that satisfy desired property profiles under feasibility constraints. While recent generative models provide continuous latent representations of chemical space, targeted optimization within these representations often leads to degraded validity, loss of structural fidelity, or unstable behavior. We introduce **MoltenFlow**, a modular framework that combines property-organized latent representations with flow-matching generative priors and gradient-based guidance. This formulation supports both conditioned generation and local optimization within a single latent-space framework. We show that guided latent flows enable efficient multi-objective molecular optimization under fixed oracle budgets with controllable trade-offs, while a learned flow prior improves unconditional generation quality.

1 INTRODUCTION

Molecular and materials discovery problems are increasingly framed as inverse design tasks: rather than evaluating a fixed set of candidates, the goal is to identify molecular structures that satisfy specified property profiles under physical and feasibility constraints. This formulation arises naturally across applications such as drug design, solvent discovery, and functional materials engineering, where target properties are expensive to evaluate and often exhibit non-trivial trade-offs. A common approach combines large-scale enumeration with learned property predictors to enable rapid virtual screening. However, such screening-based pipelines scale poorly with chemical space size and offer limited support for systematic refinement of promising candidates (Gómez-Bombarelli et al., 2018; Sindt & Rognan, 2026).

Recent advances in generative modeling have motivated a shift toward learning continuous representations of chemical space that can be directly sampled or optimized. Variational autoencoders (VAEs) and related latent variable models provide smooth latent spaces in which discrete molecular structures are embedded, enabling interpolation and gradient-based traversal (Gómez-Bombarelli et al., 2018; Jin et al., 2019; Kusner et al., 2017). Despite their success in unconditional generation, these representations are typically learned to optimize reconstruction fidelity and distributional regularization, rather than alignment with task-specific objectives. As a result, naïve latent-space optimization often leads to invalid decodings, unstable property trajectories, or collapsed diversity when strong gradients are applied (Tripp et al., 2020; Eckmann et al., 2022).

To address these limitations, a broad class of goal-directed molecular design methods has emerged. Reinforcement learning approaches formulate molecular generation as a sequential decision process (Olivecrona et al., 2017; Zhou et al., 2019), while Bayesian optimization techniques operate over learned latent spaces (Griffiths et al., 2020; Tripp et al., 2020). Although effective in specific settings, these methods often suffer from sample inefficiency, sensitivity to reward shaping, or difficulties maintaining diversity under aggressive optimization. More recently, diffusion-based generative models have demonstrated strong performance in molecular generation (Hoogeboom et al., 2022; Xu et al., 2022), and classifier-guided or conditional variants offer principled mechanisms for steering samples toward desired objectives. However, applying such guidance directly in high-

*Equal Contribution

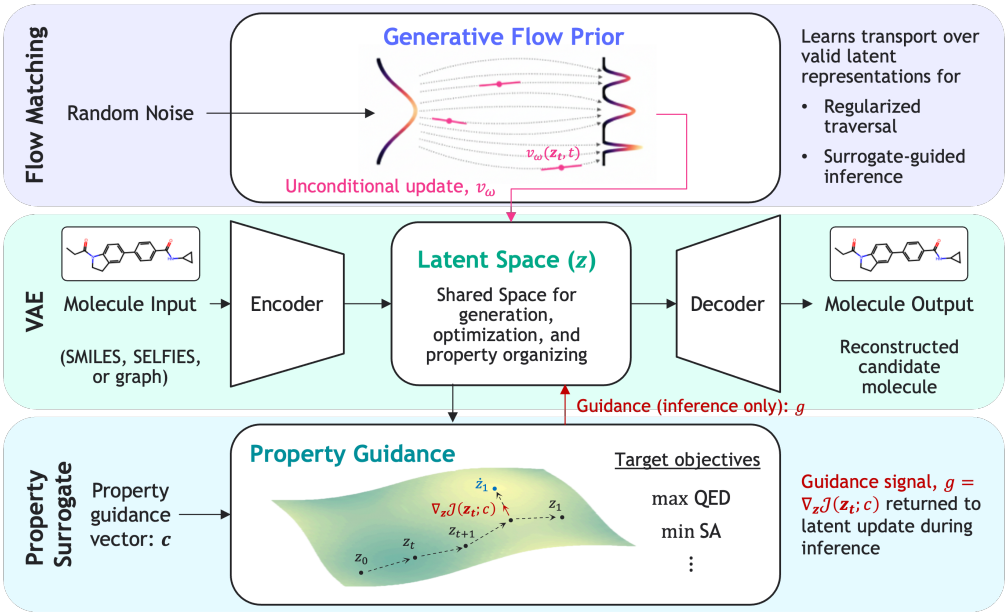


Figure 1: **MoltenFlow overview.** A VAE maps discrete molecular inputs x to continuous latent representations z and reconstructs molecules via a decoder. The latent space is *organized* through auxiliary property prediction: a differentiable property surrogate $f_\psi(z)$ induces an objective $\mathcal{J}(z; c)$ (e.g., target values or directional goals specified by a guidance vector c), and provides a guidance signal $g = \nabla_z \mathcal{J}(z_t; c)$ during inference. In parallel, a flow-matching model learns a time-dependent vector field $v_\omega(z, t)$ that transports samples from a simple base distribution (random noise) to the empirical distribution of valid latent representations. *The flow-based latent transport visualization is adapted from Sabour et al. (2025), and the property-guidance schematic is inspired by ChemFlow (Wei et al., 2024).*

dimensional molecular spaces remains computationally intensive and challenging to control (Lin et al., 2025; Chung et al., 2023).

In parallel, flow-based and diffusion-based generative models have been proposed as alternatives for learning smooth generative dynamics over molecular spaces Lipman et al. (2023); Albergo et al. (2025); Hoogeboom et al. (2022); Xu et al. (2022). Recent frameworks such as ChemFlow and PropMolFlow learn energy functions or vector fields that directly condition the generative dynamics on target properties Wei et al. (2024); Zeng et al. (2025), while methods such as MolGuidance introduce guidance terms during sampling in data space Jin et al. (2025). Other approaches, including LIMO and related latent-gradient methods, optimize properties by performing gradient-based updates in a fixed latent space, without an explicit generative prior Eckmann et al. (2022). Complementary lines of work in latent-space Bayesian optimization, such as LOL-BO and NF-BO, achieve sample efficiency through probabilistic surrogate modeling but require fitting additional models over high-dimensional latent spaces Maus et al. (2023); Lee et al. (2025).

In this work, we introduce **MoltenFlow**, a framework that occupies a distinct point in this design space. Rather than conditioning generative dynamics directly or relying on a second surrogate over latent space, MoltenFlow combines representation-level property organization with a learned latent flow prior to support controllable inverse molecular design. This design targets the regime where optimization must be both sample-efficient and stable, enabling systematic advancement of multi-objective trade-offs while remaining close to the learned data manifold. Across multi-objective molecular optimization tasks, we show that guided latent flows advance Pareto fronts while exposing an explicit trade-off between objective improvement and structural fidelity.

2 METHODS

We present **MoltenFlow**, a modular framework for molecular generation and optimization in continuous latent space. We build on latent variable generative modeling, where a VAE is used to learn a continuous representation that captures the manifold of valid chemical compounds, such that nearby latent representations correspond to structurally similar molecules. To make representations more suitable for inverse design, we further organize the latent space through auxiliary property prediction, allowing gradients from property supervision to reshape the encoder. In parallel, we learn a generative prior over latent space using flow matching, which models the distribution of valid latent representations and regularizes traversal to remain near the data manifold.

This combination produces a latent space and traversal method that is both property-aware and generative, enabling two complementary inference modes: *conditioned generation* and *optimization*. In conditioned generation mode, molecular candidates are produced by transforming noise into valid latent representations guided by desired property objectives. In optimizer mode, existing molecules are encoded, perturbed with noise, and then guided toward improved latent representations before decoding. In both cases, property objectives may be specified as explicit targets or as directional objectives, such as maximizing or minimizing a property of interest. Compared to prior latent-space optimization and reinforcement learning approaches, this formulation yields stable continuous-time guidance dynamics with an explicit guidance parameter that trades off objective improvement against remaining near the learned latent data manifold, without requiring sequential decision-making or task-specific policy training.

2.1 LATENT REPRESENTATION LEARNING

Let $x \in \mathcal{X}$ denote a molecular structure represented in a discrete format (e.g., SMILES or SELFIES; see Appendix B for representation details). We learn a continuous latent representation z using a variational autoencoder (VAE). Here, z is a set of token-level latent representations $z \in \mathbb{R}^{K\times d}$ produced by attention-based pooling over encoder outputs. When a vector-valued latent representation is required, we denote the pooled representation as $\bar{z} \in \mathbb{R}^d$. The generative model is defined by a prior $p(z)$ and a decoder $p_\theta(x \mid z)$, while inference is performed using a variational posterior $q_\phi(z \mid x)$, where θ and ϕ parameterize the decoder and encoder, respectively. Model parameters are learned by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}_{\mathrm{VAE}}(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z)) \tag{1}$$

We parameterize $q_\phi(z \mid x)$ as a diagonal Gaussian and use the reparameterization trick, $z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, to optimize this objective by backpropagation.

This objective encourages latent representations that support accurate reconstruction while remaining close to the prior distribution. However, this representation is optimized primarily for reconstruction fidelity and distributional regularization; it does not explicitly encode task-relevant semantics.

We denote the induced latent data distribution (aggregated posterior) by $q_\phi(z) = \mathbb{E}_{x\sim p_{\mathrm{data}}}\, q_\phi(z \mid x)$. In later stages, we treat samples $z_1 \sim q_{\phi'}(z \mid x)$ from the training set as samples from the target latent distribution $p_1(z) = q_{\phi'}(z)$, where ϕ' denotes the encoder parameters after property-oriented fine-tuning (Section 2.2).

2.2 PROPERTY-ORIENTED LATENT SPACE TRANSFORMATION

To make latent space traversal meaningful for inverse design, we augment latent representation learning with property supervision. Let $y \in \mathbb{R}^P$ denote a vector of molecular properties of interest, and let $f_\psi : \mathbb{R}^{K\times d} \to \mathbb{R}^P$ be a differentiable surrogate model that predicts properties from latent representations.

We introduce an auxiliary property prediction loss

$$\mathcal{L}_{\mathrm{prop}}(z, y) = \ell(f_\psi(z),\, y) \tag{2}$$

where $\ell(\cdot, \cdot)$ is an appropriate regression loss. During training, this loss is combined with the VAE objective to yield

$$\mathcal{L}(x, y) = \mathcal{L}_{\mathrm{VAE}}(x) + \lambda\, \mathcal{L}_{\mathrm{prop}}(z, y) \tag{3}$$

with $z \sim q_\phi(z \mid x)$ and $\lambda \geq 0$ controlling the strength of property supervision.

Allowing gradients of $\mathcal{L}_{\text{prop}}$ to backpropagate through the encoder q_ϕ reshapes the latent geometry such that local directions in latent space correlate with changes in predicted properties Gómez-Bombarelli et al. (2018). This process can be interpreted as transforming the latent representation into one that is better aligned with downstream optimization objectives. The resulting latent space remains continuous and smooth, but is additionally structured by property-relevant variation.

2.3 Latent Flow Matching

While property-oriented latent representations improve local interpretability, they do not guarantee that latent trajectories remain within realistic regions of chemical space. Syntactically robust representations such as SELFIES ensure valid decoding but do not prevent drift into low-density regions; we therefore learn a flow-based prior over latent space to model the empirical distribution and regularize trajectories toward the data manifold.

Flow matching seeks to learn a time-dependent vector field

$$v_\omega : (\mathbb{R}^{K\times d} \times [0,1]) \to \mathbb{R}^{K\times d}, \tag{4}$$

where ω denotes the parameters of the flow model, which transports samples from a simple base distribution $p_0(z)$ (e.g., Gaussian noise) to a target distribution $p_1(z)$ (the empirical distribution of latent codes). Training proceeds by sampling endpoints $z_0 \sim p_0$, $z_1 \sim p_1$, and a time $t \sim \mathcal{U}[0,1]$. For notational convenience, we write $z_t := z(t)$ to denote the latent state at time $t \in [0,1]$. We use the linear probability path

$$z_t = (1-t)z_0 + tz_1, \qquad \frac{dz_t}{dt} = z_1 - z_0 \tag{5}$$

Flow matching then fits the time-dependent vector field by regression:

$$\mathcal{L}_{\text{FM}}(\omega) = \mathbb{E}_{z_0\sim p_0,\, z_1\sim p_1,\, t\sim\mathcal{U}[0,1]} \left[\|v_\omega(z_t,t) - (z_1 - z_0)\|_2^2 \right] \tag{6}$$

The learned vector field captures the geometry of the latent data distribution, providing a manifold-aware generative model. Sampling from this flow yields latent representations that remain close to regions supported by training data, while avoiding the need for explicit likelihood computation or invertibility constraints.

At inference time, the learned prior samples latents by integrating the ODE $\dot{z}(t) = v_\omega(z(t),t)$ from $t = 0$ to 1 with initial condition $z(0) \sim p_0$; the final latent $z(1)$ is decoded to a molecule using the VAE decoder.

2.4 Guided Latent Dynamics for Generation and Optimization

The final component of MoltenFlow is a guidance mechanism that combines the learned latent flow with gradients derived from surrogate objectives. Let $f_\psi(z) \in \mathbb{R}^P$ denote the surrogate-predicted properties associated with a latent representation z, and let c denote a vector specifying desired targets or objective preferences for these properties. We define a scalar objective function $\mathcal{J}(z;c)$, parameterized by c, and compute its gradient with respect to latent space:

$$g(z) = \nabla_z \mathcal{J}(z;c) \tag{7}$$

We interpret $\mathcal{J}(z;c)$ as a loss function to be minimized; maximization objectives are handled by defining $\mathcal{J}$ with an appropriate negative sign (see Appendix C). During inference, we modify the latent flow dynamics by combining the generative velocity with the objective gradient:

$$\dot{z}(t) = v_\omega(z(t),t) - \gamma\, g(z(t)), \tag{8}$$

where $\gamma > 0$ controls the strength of guidance. This formulation yields a continuous-time dynamical system that balances two competing effects: movement along the learned data manifold and movement toward improved objective values specified by c. Using a learned latent prior in Eq. 8 is critical: direct gradient ascent on surrogate objectives often pushes latents into low-density regions

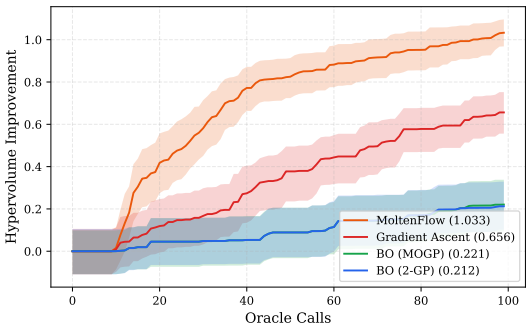


Figure 2: **Hypervolume improvement (HVI)** as a function of oracle calls under a fixed budget. Curves show mean performance across random seeds, with 90% bootstrap confidence intervals.

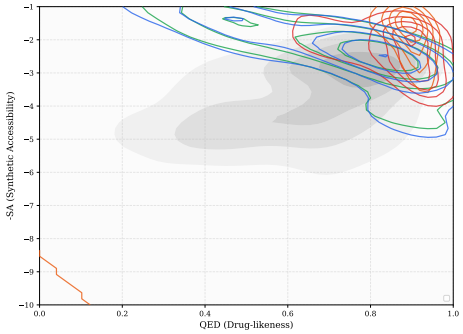


Figure 3: **Densities of Pareto fronts** after budgeted optimization. MoltenFlow yields dense, stable fronts, while BO baselines yield sparser, more variable fronts.

where decoding becomes unreliable. The flow term $v_\omega(z, t)$ regularizes these updates by biasing trajectories toward the support of $q_{\phi'}(z)$, improving feasibility for a given guidance strength γ.

By varying the form of $\mathcal{J}$, the conditioning vector c, and the initial condition $z(0)$, the same guided dynamics support multiple inference modes. Starting from samples drawn from the base distribution yields generation directed toward specified property objectives, while initializing from the latent encoding of an existing molecule enables local optimization. In both cases, the learned latent flow regularizes traversal, ensuring that optimization proceeds through regions of latent space associated with valid molecular structures.

3 Experiments

3.1 Multi-objective Molecular Optimization

We evaluate MoltenFlow's optimizer mode through two complementary studies that assess performance in simulated practical discovery workflows and advancing global Pareto fronts. Unless otherwise noted, all optimization experiments use a fixed MoltenFlow configuration selected via Bayesian hyperparameter optimization, operate on the same learned latent representation with a shared surrogate model, and employ SELFIES to ensure syntactic validity and isolate optimization dynamics. Details of the search space and tuning procedure are provided in Appendix E.1.

3.1.1 Budgeted optimization

We first evaluate MoltenFlow in a budgeted optimization setting that mirrors practical molecular discovery workflows, where candidate evaluations are expensive and optimization must proceed under a fixed oracle budget. In this regime, MoltenFlow consistently achieves larger and faster Pareto-front improvements across seeds than both latent-space Bayesian optimization and unregularized gradient-based optimization, while maintaining perfect validity. This advantage emerges early in the optimization process and persists across random initializations, highlighting the utility of combining surrogate gradients with a learned, manifold-aware latent flow.

Each optimization run begins from a small set of randomly selected molecules and proceeds sequentially, proposing new candidates until the oracle budget is exhausted (see Appendix E.2 for details). We compare MoltenFlow against two latent-space Bayesian optimization baselines: one with a multi-output Gaussian process and the other with independent Gaussian processes per objective, both with qEHVI acquisition. We also compare against a gradient ascent ablation that removes the flow prior and follows surrogate gradients directly. All methods are evaluated under the same fixed oracle budget; MoltenFlow leverages a pretrained surrogate (offline setting), while Bayesian optimization fits its surrogate online.

Figure 2 illustrates the evolution of hypervolume improvement as a function of oracle calls. MoltenFlow exhibits rapid early gains, reaching a substantially improved Pareto front within a small

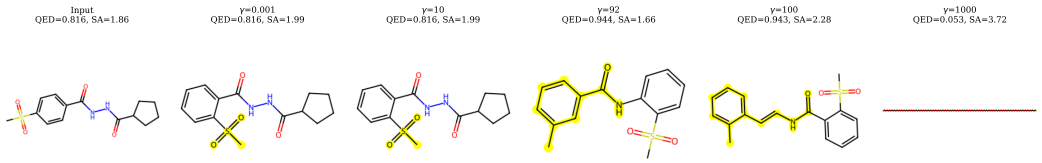


Figure 4: **Qualitative effect of guidance strength on optimization.** As the guidance strength γ increases, MoltenFlow produces progressively larger structural changes. Small γ yields minimal edits dominated by the latent flow prior, intermediate γ induces coherent improvements in QED with reasonable SA, and large γ leads to aggressive modifications indicative of over-optimization.

fraction of the available budget. In contrast, both Bayesian optimization variants improve more slowly and exhibit greater variability across runs, consistent with the challenges of fitting Gaussian process models in moderate-dimensional latent spaces. The gradient ascent ablation performs competitively at early stages but consistently underperforms MoltenFlow, indicating that flow-based regularization contributes meaningfully beyond the surrogate gradient direction alone.

The qualitative structure of the optimized solution sets further distinguishes the methods. As shown in Figure 3, MoltenFlow concentrates probability mass in the high-QED, low-SA region of objective space, producing dense and stable Pareto fronts across random seeds. Bayesian optimization methods, by contrast, yield sparser fronts with higher variance, reflecting sensitivity to acquisition noise and surrogate uncertainty under limited budgets.

From a computational perspective, MoltenFlow is substantially more efficient than Bayesian optimization (see Appendix E.2.5). By directly leveraging gradients from a differentiable surrogate and avoiding repeated Gaussian process fitting and acquisition maximization, MoltenFlow achieves comparable or better improvements with significantly lower wall-clock time per run. This efficiency advantage becomes increasingly relevant as oracle budgets grow or optimization is embedded within iterative discovery pipelines.

While the budgeted setting highlights MoltenFlow's practical efficiency under limited evaluations, we next examine how guided latent flows systematically reshape the Pareto front itself and the trade-offs induced by varying the guidance strength.

3.1.2 Pareto front advancement

We study MoltenFlow's ability to advance the Pareto front for drug-like molecules by jointly optimizing Quantitative Estimate of Drug-likeness (QED; higher is better) and Synthetic Accessibility (SA; lower is better) on ZINC250K. Guided latent flows induce a clear and interpretable trade-off regime: for a well-chosen range of guidance strengths, MoltenFlow substantially expands the Pareto front while preserving molecular validity and scaffold diversity, whereas overly aggressive guidance leads to distributional drift and structural collapse.

Figure 4 provides a qualitative view of this behavior. At very small guidance strengths, optimized molecules remain very similar to their inputs, reflecting dominance of the learned latent flow prior. As guidance increases, structural changes emerge gradually and coherently, improving QED without immediately sacrificing synthesizability. In contrast, excessively large guidance produces abrupt edits that compromise structural integrity, consistent with over-optimization beyond the surrogate's reliable regime.

A differentiable surrogate model predicts QED and SA from latent representations and provides gradients used to guide optimization. Optimization begins from latent encodings of molecules near the empirical Pareto front, injects modest Gaussian noise to encourage local exploration, and integrates guided dynamics that combine the learned latent flow velocity with normalized property gradients. The only degree of freedom varied in this study is the guidance strength parameter γ, which controls the relative influence of the surrogate objective versus the generative flow prior.

Across a wide sweep of γ values, a consistent pattern emerges (Figure 5 and 6). At low γ, trajectories remain close to the data manifold, yielding negligible Pareto expansion but near-perfect scaffold preservation. As γ increases into an intermediate regime, MoltenFlow reliably advances

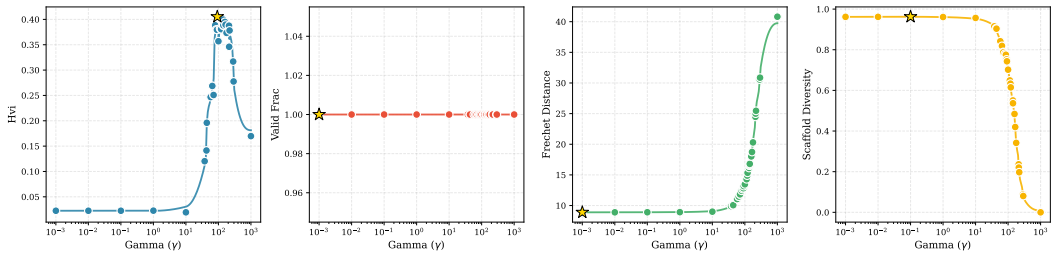
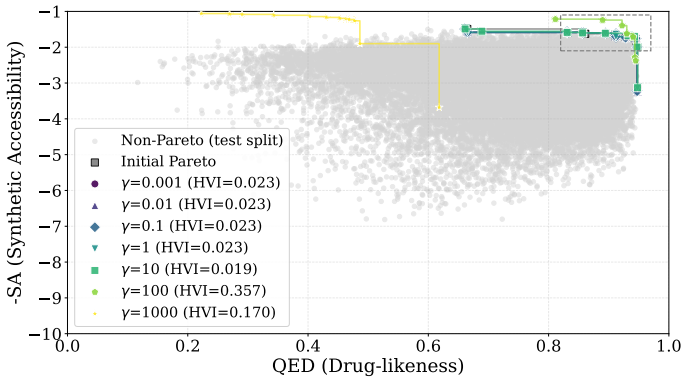
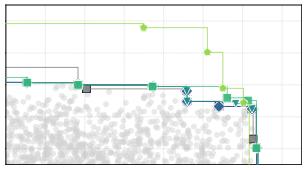


Figure 5: **Effect of guidance strength on optimization and distributional properties.** Each panel shows a metric as the guidance strength γ increases (log scale). From left to right: hypervolume improvement (HVI), validity, Fréchet distance (FD-FP), and scaffold diversity. HVI increases sharply in an intermediate regime, indicating effective Pareto-front advancement, while excessive guidance leads to rising Fréchet distance and collapsing diversity, signaling over-optimization. Starred points denote the γ achieving the best value for each metric.





γ	HVI (%)↑	Scaf. Div.↑	FD-FP↓
0.001	0.39	0.962	8.9
0.1	0.39	0.963	8.9
1	0.39	0.961	8.9
10	0.33	0.956	9.0
100	**6.13**	0.702	13.5
1000	2.91	0.000	40.8

Figure 6: **Pareto-front advancement and guidance-strength effects.** (**Left**) Optimized Pareto fronts obtained at different guidance strengths γ. At intermediate γ (green), MoltenFlow produces fronts that strictly dominate the original, expanding into the high-QED, low-SA region while retaining a broad set of trade-offs. (**Right**) Zoomed view of the high-QED/low-SA corner (top) and summary metrics across guidance strengths (bottom), illustrating the trade-off between hypervolume improvement, scaffold diversity, and distributional shift (see Appendix E.3.4 for metric definitions).

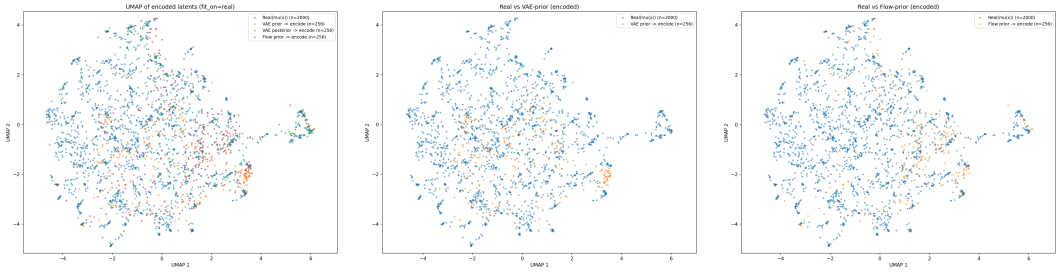
the Pareto front into regions of higher QED and lower SA, achieving the largest hypervolume gains while maintaining substantial scaffold diversity. Beyond this regime, further increasing γ degrades performance: diversity collapses and distributional metrics indicate significant departure from the training manifold.

Figure 6 visualizes the resulting Pareto fronts. At the optimal guidance strength, the optimized front strictly dominates the original test-set front, expanding into the high-QED, low-SA region that corresponds to more attractive drug-like candidates. Importantly, this expansion is not achieved by collapsing to a narrow set of solutions, but by producing a broader front that reflects meaningful trade-offs between objectives.

Taken together, these results demonstrate that MoltenFlow enables controlled and interpretable multi-objective optimization. The guidance strength parameter provides a continuous knob between conservative, scaffold-preserving refinement and more aggressive exploration, making explicit the trade-offs inherent in latent-space molecular optimization.

3.2 Molecular Generation and Ablations

We next examine MoltenFlow in a pure generation setting, with surrogate guidance disabled at inference. These experiments are not intended to demonstrate state-of-the-art unconditional generation



(a) Overlay (fit on real). (b) Real vs VAE prior. (c) Real vs flow prior.

Figure 7: **UMAP of posterior-mean latent embeddings** $\mu_\phi(x)$ for real molecules (fit set) and for generated molecules after re-encoding. Flow-prior samples align more tightly with the support of real embeddings compared to VAE-prior samples, consistent with improved validity and reduced FID-FP in Table 1.

performance, but rather to isolate the contribution of two design choices: learning a flow-based latent prior and selecting an appropriate molecular string representation. The central takeaway is that a learned latent flow prior produces samples that better align with the support of the training distribution than a standard VAE prior, and that this effect is amplified by using a syntactically robust representation such as SELFIES.

3.2.1 Flow matching ablation

We begin by comparing unconditional sampling from a standard VAE Gaussian prior against sampling from a learned flow-matching prior, holding the decoder and training data fixed. In both cases, latent samples are decoded without surrogate guidance ($\gamma = 0$), allowing us to attribute differences in generation quality directly to the choice of latent prior. Table 1 summarizes the resulting generation statistics. Replacing the Gaussian prior with a flow-matching prior improves validity and substantially reduces distributional discrepancy, as measured by fingerprint Fréchet distance and descriptor-level KL divergence. Uniqueness and novelty among valid samples remain high in both settings, indicating that gains are not achieved by mode collapse. For reference, we also report reconstruction from posterior means, which serves as an approximate upper bound on validity but does not constitute an unconditional generator.

The structure of the generated samples in latent space further clarifies these trends. Figure 7 visualizes posterior-mean latent embeddings of real molecules alongside embeddings of generated samples after re-encoding. Samples drawn from the flow prior concentrate more tightly within the support of the real-data manifold, whereas samples from the VAE prior spread into low-density regions. This qualitative alignment is consistent with the improvements in validity and distributional metrics observed in Table 1, and supports the role of the flow prior as a manifold-aware regularizer for latent sampling.

3.2.2 Molecular representation ablation

We then investigate how the choice of molecular representation interacts with the learned flow prior. Using the same VAE+flow architecture, we compare models trained on SMILES strings against those trained on SELFIES, again sampling unconditionally from the flow prior. As shown in Table 1, SELFIES yields perfect syntactic validity while preserving high uniqueness and novelty among generated molecules. Scaffold diversity remains comparable across representations, while distributional metrics reveal a modest trade-off: SELFIES improves descriptor-level KL divergence but exhibits slightly higher fingerprint Fréchet distance. This behavior reflects the well-known tension between enforcing syntactic constraints and matching fine-grained distributional statistics.

Figure 11 provides a latent-space perspective on this comparison. For both representations, flow-prior samples re-encode to regions overlapping with the real-data manifold, but SELFIES exhibits tighter coverage and fewer outliers. Importantly, these differences arise despite identical training procedures and latent dynamics, underscoring the impact of representation choice on the effective support of the generative model.

Table 1: **Unconditional generation ablations (no surrogate guidance). A1** compares the VAE prior vs. flow-matching (FM) prior under SMILES. **A2** compares SMILES vs. SELFIES under the FM prior. FD-FP is Fréchet distance in fingerprint embedding space; Avg KL is the mean KL divergence over RDKit descriptors.

Rep.	Prior	Valid(%)↑	Unique(%)↑	Novel(%)↑	Scaf. div.↑	Avg KL↓	FD-FP↓
SMILES	VAE prior	5.76	98.44	99.47	0.944	0.299	13.23
SMILES	Flow prior (FM)	9.56	97.91	99.04	0.700	0.191	8.15
SELFIES	Flow prior (FM)	100.00	99.07	99.66	0.694	0.161	10.72
SMILES	VAE post. ($\mu_\phi(x)$; ref.)	26.07	78.02	71.58	0.438	0.207	4.67

While unconditional generation is not the primary focus of MoltenFlow, these results provide essential context for the optimizer-mode experiments, where the same flow prior regularizes guided trajectories and helps prevent drift away from the data manifold.

4 Conclusion

We introduced **MoltenFlow**, a latent-space framework for molecular generation and optimization that combines property-oriented latent space organization, flow-based generative modeling, and surrogate-based guidance. By unifying generation and optimization within a single latent-space formulation, MoltenFlow enables stable and controllable multi-objective molecular design.

Empirically, we evaluated MoltenFlow on multi-objective optimization tasks derived from ZINC250K, focusing on maximizing drug-likeness (QED) while minimizing synthetic accessibility (SA). Across both budgeted optimization and Pareto-front advancement experiments, MoltenFlow achieved faster and larger Pareto improvements than latent-space Bayesian optimization and unregularized gradient ascent under identical oracle budgets. The guidance strength parameter γ provided a clear and interpretable control over the trade-off between property improvement and structural fidelity, with an intermediate regime yielding substantial gains before over-optimization. In unguided generation, learning a latent flow prior improved validity and distributional alignment relative to the VAE prior, with SELFIES further ensuring syntactic robustness.

Across these studies, MoltenFlow demonstrates that guided latent flows can reliably improve multi-objective performance while providing explicit control over optimization aggressiveness. Rather than enforcing validity or diversity implicitly, the framework exposes these trade-offs through interpretable parameters, enabling principled navigation of the latent design space. These results indicate that combining property-organized latent representations with learned generative priors can substantially improve the reliability and transparency of latent-space molecular optimization.

Limitations. MoltenFlow relies on surrogate property models and is therefore sensitive to surrogate accuracy and calibration, particularly under strong guidance. The current formulation operates on 2D molecular representations and does not capture three-dimensional structure or conformational effects. Performance is also sensitive to inference-time hyperparameters such as integration start time and noise scale. Additionally, our evaluation focuses on two proxy objectives (QED and SA), which do not capture the full complexity of molecular design tasks. Finally, while MoltenFlow provides controlled local optimization dynamics, it does not offer formal guarantees of global optimality.

Future Work. Incorporating uncertainty-aware or ensemble-based surrogate models could improve robustness and enable risk-sensitive optimization. Adaptive schemes for selecting integration start time and noise scale—potentially informed by local latent density, surrogate uncertainty, or trajectory diagnostics—could further reduce sensitivity to inference-time hyperparameters. Extending the framework to incorporate additional molecular properties beyond QED and SA, three-dimensional representations, and physics-informed objectives would broaden applicability to structure-sensitive design tasks. Evaluating on materials science benchmarks and integrating with experimental feedback loops are also important directions that could enable closed-loop discovery pipelines that combine latent optimization with real-world validation.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://arxiv.org/abs/1907.10902.

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2025. URL https://arxiv.org/abs/2303.08797.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization, 2020. URL https://arxiv.org/abs/1910.06403.

Guy W. Bemis and Mark A. Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of Medicinal Chemistry*, 39(15):2887–2893, 1996. doi: 10.1021/jm9602928. URL https://doi.org.

George R. Bickerton, Giulio V. Paolini, Jérôme Besnard, Sorel Muresan, and Andrew L. Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, January 2012. doi: 10.1038/nchem.1243.

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. ISSN 2169-3536. doi: 10.1109/access.2020.2990567. URL http://dx.doi.org/10.1109/ACCESS.2020.2990567.

Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. doi: 10.48550/arXiv.1806.07366. URL https://arxiv.org/abs/1806.07366.

Hyungjin Chung, Jeongsol Kim, Michael T. McCann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations (ICLR)*, 2023. doi: 10.48550/arXiv.2209.14687. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space, 2023. URL https://arxiv.org/abs/2307.08698.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement, 2021. URL https://arxiv.org/abs/2105.08195.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021.

Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation, 2024. URL https://arxiv.org/abs/2404.19739.

Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K. Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation, 2022. URL https://arxiv.org/abs/2206.09010.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, New York, 1993. ISBN 0412042312.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1(1):8, June 2009. doi: 10.1186/1758-2946-1-8.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019. doi: 10.48550/arXiv.1810.01367. URL https://openreview.net/forum?id=rJxgknCcK7.

Ryan-Rhys Griffiths et al. Constrained bayesian optimization for automatic chemical design. *Chemical Science*, 2020.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Emiel Hoogeboom et al. Equivariant diffusion for molecule generation in 3d. *ICML*, 2022.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 754–762, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/hutter14.html.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012. doi: 10.1021/ci3001277. URL https://doi.org/10.1021/ci3001277.

Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. Efficient 3d molecular generation with flow matching and scale optimal transport, 2024. URL https://arxiv.org/abs/2406.07266.

Jirui Jin, Cheng Zeng, Pawan Prakash, Ellad B. Tadmor, Adrian Roitberg, Richard G. Hennig, Stefano Martiniani, and Mingjie Liu. Molguidance: Advanced guidance strategies for conditional molecular generation with flow matching, 2025. URL https://arxiv.org/abs/2512.12198.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2019. URL https://arxiv.org/abs/1802.04364.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100 *Machine Learning: Science and Technology*, 1 (4):045024, October 2020. ISSN 2632-2153. doi: 10.1088/2632-2153/aba947. URL http://dx.doi.org/10.1088/2632-2153/aba947.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder, 2017. URL https://arxiv.org/abs/1703.01925.

Greg Landrum. *RDKit: Open-source cheminformatics*. URL https://www.rdkit.org/.

Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J. Kim. Latent bayesian optimization via autoregressive normalizing flows, 2025. URL https://arxiv.org/abs/2504.14889.

Haowei Lin, Shanda Li, Haotian Ye, Yiming Yang, Stefano Ermon, Yitao Liang, and Jianzhu Ma. Tfg-flow: Training-free guidance in multimodal generative flow, 2025. URL https://arxiv.org/abs/2501.14216.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023. doi: 10.48550/arXiv.2209.03003. URL https://openreview.net/forum?id=XVjTT1nw5z.

Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

Natalie Maus, Haydn T. Jones, Juston S. Moore, Matt J. Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space bayesian optimization over structured inputs, 2023. URL https://arxiv.org/abs/2201.11872.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning. *Journal of Cheminformatics*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2021. URL https://arxiv.org/abs/1912.02762.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation, 2025. URL https://arxiv.org/abs/2506.14603.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *ICANN*, 2018.

François Sindt and Didier Rognan. Structure-based virtual screening of ultra-large chemical spaces: Advances and pitfalls. *European Journal of Medicinal Chemistry*, 305:118576, 2026. ISSN 0223-5234. doi: https://doi.org/10.1016/j.ejmech.2026.118576. URL https://www.sciencedirect.com/science/article/pii/S0223523426000218.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. doi: 10.48550/arXiv.2011.13456. URL https://arxiv.org/abs/2011.13456.

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. doi: 10.48550/arXiv.2302.00482. URL https://openreview.net/forum?id=CD9Snc73AW.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining, 2020. URL https://arxiv.org/abs/2006.09191.

Guanghao Wei, Yining Huang, Chenru Duan, Yue Song, and Yuanqi Du. Navigating chemical space with latent flows, 2024. URL https://arxiv.org/abs/2405.03987.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

Minkai Xu et al. Geodiff: A geometric diffusion model for molecular conformation generation. *ICLR*, 2022.

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation, 2019. URL https://arxiv.org/abs/1806.02473.

Chengxi Zang and Fei Wang. Moflow: An invertible flow model for generating molecular graphs. *ICML*, 2020.

Cheng Zeng, Jirui Jin, George Karypis, Mark Transtrum, Ellad B. Tadmor, Richard G. Hennig, Adrian Roitberg, Stefano Martiniani, and Mingjie Liu. Propmolflow: Property-guided molecule generation with geometry-complete flow matching, 2025. URL https://arxiv.org/abs/2505.21469.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):10752, 2019. doi: 10.1038/s41598-019-47148-x. URL https://doi.org/10.1038/s41598-019-47148-x.

A Extended Related Work

A.1 Molecular representations and validity.

String-based molecular representations such as SMILES (Weininger, 1988) are compact but prone to syntactic invalidity. SELFIES provides a robust alternative that guarantees syntactic validity by construction (Krenn et al., 2020). Dimensionality reduction techniques such as UMAP are commonly used to visualize latent-space structure and assess coverage (McInnes et al., 2018).

A.2 Molecular latent representation learning.

A central theme in molecular generative modeling is the use of continuous latent representations to capture chemical variation in a form amenable to interpolation and optimization. Early work with SMILES-based variational autoencoders demonstrated that latent spaces learned from molecular strings encode meaningful structure and permit gradient-based exploration Gómez-Bombarelli et al. (2018); Kusner et al. (2017). Subsequent extensions incorporated graph-based representations and structured decoders to improve validity and expressiveness Jin et al. (2019); Simonovsky & Komodakis (2018). Despite these advances, latent representations learned purely for reconstruction are not inherently organized for inverse design, and unconstrained traversal in such spaces often leads to invalid molecules or degraded diversity when aggressively optimizing properties.

A.3 Goal-directed molecular optimization.

A large body of work addresses the problem of generating molecules that optimize one or more target properties. Reinforcement learning approaches formulate molecule construction as a sequential decision process, allowing direct optimization of reward functions Olivecrona et al. (2017); Zhou et al. (2019). While flexible, these methods are sensitive to reward shaping and frequently require careful tuning to avoid mode collapse. Bayesian optimization has also been applied in learned latent spaces as a sample-efficient alternative Griffiths et al. (2020); Tripp et al. (2020), with recent methods such as LOL-BO introducing trust-region strategies to stabilize search Maus et al. (2023). However, latent-space BO methods typically rely on fitting an additional surrogate over the latent space and can struggle with sample efficiency as latent dimensionality increases.

A.4 Diffusion and flow-based generative models.

Diffusion-based models have achieved state-of-the-art performance in molecular generation, particularly when incorporating geometric inductive biases (Hoogeboom et al., 2022; Xu et al., 2022). From a broader perspective, diffusion can be understood through the score-based SDE framework, which unifies stochastic and deterministic sampling views and clarifies how guidance terms influence reverse-time dynamics (Ho et al., 2020; Song et al., 2021). Classifier-guided and conditional diffusion techniques provide principled mechanisms for steering generation toward desired properties (Dhariwal & Nichol, 2021), but applying diffusion directly in molecular space is computationally expensive and controlling multi-objective optimization trajectories remains challenging.

A.5 Flow matching as a generative prior.

Flow-based generative models provide a complementary approach, learning continuous-time dynamics that transport samples from simple base distributions to complex target distributions Rezende & Mohamed (2016); Papamakarios et al. (2021). Recent work on flow matching reframes this process as learning vector fields along predefined interpolation paths, avoiding explicit likelihood computation Lipman et al. (2023); Albergo et al. (2025). Applying flow matching in the latent space of a pretrained autoencoder yields a scalable analogue to latent diffusion Dao et al. (2023). Several recent studies have applied flow matching to molecular generation, including MoFlow, FlowMol, SemlaFlow, and related frameworks Zang & Wang (2020); Dunn & Koes (2024); Irwin et al. (2024), with an emphasis on unconditional sampling or efficient generation rather than guided optimization.

A.6 NORMALIZING FLOWS AND CONTINUOUS-TIME MODELS.

Normalizing flows provide exact likelihood-based generative models through invertible transformations (Rezende & Mohamed, 2016; Papamakarios et al., 2021). Continuous normalizing flows and neural ODEs extend this framework to continuous-time dynamics (Chen et al., 2018; Grathwohl et al., 2019). Recent variants, including rectified flow and optimal-transport-based couplings, simplify training and reduce inference cost (Liu et al., 2023; Tong et al., 2024).

A.7 GUIDANCE AND CONDITIONAL GENERATION.

Guidance mechanisms have been widely studied in diffusion and flow-based models, including classifier guidance, classifier-free guidance, and posterior sampling formulations (Dhariwal & Nichol, 2021; Chung et al., 2023; Ho & Salimans, 2022). Recent work has explored training-free guidance in multimodal settings (Lin et al., 2025). These approaches typically modify the generative dynamics directly, whereas MoltenFlow uses surrogate-based gradients to guide a latent flow prior.

A.8 POSITIONING THIS WORK.

MoltenFlow builds on these three lines of work by combining property-organized latent representations with a learned latent flow prior and surrogate-based gradient guidance. In contrast to latent-space Bayesian optimization, MoltenFlow avoids fitting a second surrogate over the latent space and instead exploits direct gradients from a differentiable property head. Relative to recent flow-matching-based molecular generators such as ChemFlow, which learn energy or vector fields over a fixed latent space Wei et al. (2024), MoltenFlow explicitly reshapes the latent representation through auxiliary property supervision and uses the learned flow to regularize guided optimization trajectories. This design enables controlled, sample-efficient multi-objective optimization while exposing an interpretable trade-off between improvement and feasibility.

B MOLECULAR STRING REPRESENTATIONS (SMILES AND SELFIES).

MoltenFlow supports discrete molecular inputs x provided as string representations. By default we use SMILES (Weininger, 1988). For the representation ablation (Section 3.2), we additionally train and sample models on SELFIES (Krenn et al., 2020), a grammar-constrained string representation designed to reduce syntactic invalidity under token-level perturbations.

B.1 TOKENIZATION.

For SMILES, we tokenize at the character (or multi-character atom) level with special tokens for padding and sequence boundaries (BOS/EOS), and truncate/pad sequences to a fixed maximum length L. For SELFIES, we tokenize by bracketed SELFIES symbols, yielding a sequence of discrete tokens with the same BOS/EOS/padding convention and the same maximum length L. All model architectures and hyperparameters are held fixed between SMILES and SELFIES runs unless stated otherwise; only the tokenizer/vocabulary changes.

B.2 CANONICALIZATION AND VALIDITY.

For evaluation, generated strings are converted to canonical SMILES using RDKit canonicalization. For SELFIES-trained models, generated SELFIES strings are first decoded into SMILES (using the standard SELFIES decoder) and then canonicalized. A sample is counted as *valid* if it successfully parses into an RDKit molecule after conversion/canonicalization; otherwise it is invalid.

C OBJECTIVE CONSTRUCTION FOR GUIDED LATENT DYNAMICS

This section provides explicit examples of how the scalar objective $\mathcal{J}(z; c)$ used for guided latent dynamics is constructed from surrogate predictions. As described in §2.4, $\mathcal{J}$ is interpreted as a loss function to be minimized, and maximization objectives are handled by appropriate sign conventions.

Target-based objectives. For objectives that specify desired target values for molecular properties, we define $\mathcal{J}$ as a weighted squared error between surrogate predictions and target values:

$$\mathcal{J}_{\text{target}}(z; c) = \sum_{i=1}^{P} w_i \left(f_{\psi,i}(z) - c_i\right)^2, \tag{9}$$

where $f_{\psi,i}(z)$ denotes the surrogate prediction for property i, c_i is the desired target value, and $w_i \geq 0$ controls the relative importance of each property. This form encourages latent updates that move predictions toward specified targets.

Directional (maximize/minimize) objectives. For directional optimization, where properties are to be maximized or minimized without explicit targets, we define

$$\mathcal{J}_{\text{dir}}(z; c) = -\sum_{i=1}^{P} s_i \, f_{\psi,i}(z), \tag{10}$$

where $s_i \in \{+1, -1\}$ encodes the optimization direction for property i. Specifically, $s_i = +1$ corresponds to maximizing property i, while $s_i = -1$ corresponds to minimizing it. With this convention, gradient descent on $\mathcal{J}_{\text{dir}}$ implements the desired maximize/minimize behavior.

Encoding objectives via c. In practice, the conditioning vector c specifies either target values (for $\mathcal{J}_{\text{target}}$) or directional indicators (for $\mathcal{J}_{\text{dir}}$), and may also include optional property weights. Both formulations are compatible with the guided dynamics in Eq. 8.

Examples: QED and SA. In the experiments, we optimize for high drug-likeness (QED) and low synthetic accessibility (SA). Using the directional formulation, this corresponds to setting

$$s_{\text{QED}} = +1, \qquad s_{\text{SA}} = -1,$$

so that gradient descent on $\mathcal{J}_{\text{dir}}$ increases QED while decreasing SA. Alternatively, target-based objectives may be constructed by specifying desired target values c_{QED} and c_{SA} in $\mathcal{J}_{\text{target}}$.

D Model architecture and training details

VAE. We use a Transformer-based sequence VAE for string inputs (SMILES/SELFIES). The encoder is a Transformer encoder with N_{enc} layers, model width d_{model}, H attention heads, and feedforward width d_{ff}. The latent interface uses attention pooling to produce K latent tokens of dimension d, parameterizing a diagonal Gaussian posterior $q_\phi(z \mid x)$. The decoder is a Transformer decoder with N_{dec} layers that conditions on z via cross-attention and is trained with teacher forcing.

Property surrogate. The property predictor f_ψ operates on the pooled latent $\bar{z}$ and is implemented as an MLP with *3* layers and hidden size *1024*, with output heads for QED and SA. We bound outputs using sigmoid activation for QED and scaled sigmoid for SA to enforce $\text{QED} \in [0, 1]$ and $\text{SA} \in [1, 10]$.

Flow model. A Transformer-based architecture with 10 layers, hidden dimension 256, 8 attention heads, feedforward dimension 512, and dropout rate 0.1 parameterizes the velocity field. Time conditioning uses a 128-dimensional sinusoidal embedding. The model is trained for 100 epochs with batch size 1024 and learning rate 2×10^{-4} using the conditional flow matching objective with optimal transport coupling (Lipman et al., 2023; Tong et al., 2024).

Training protocol. Stage (i) VAE pretraining uses objective Eq. 1 for 150 epochs with optimizer AdamW, learning rate $1e-4$, batch size 256. Stage (ii) property fine-tuning minimizes Eq. 2 with $\lambda = 1$ for 150 epochs. Stage (iii) flow matching minimizes Eq. 6 using $z_0 \sim \mathcal{N}(0, I)$ and $z_1 \sim q_{\phi'}(z \mid x)$ for 300 epochs, with learning rate $2e-4$ and batch size 1024. For each stage, the epoch with the best validation loss is selected.

E EXPERIMENTS DETAILS

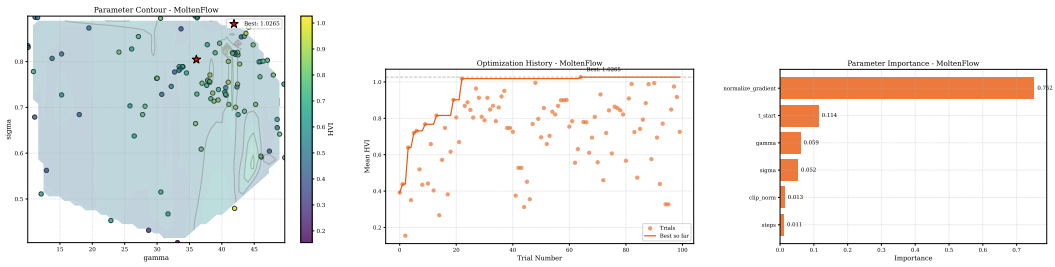
E.1 HYPERPARAMETER OPTIMIZATION FOR MOLTENFLOW

MoltenFlow hyperparameters were selected via Bayesian hyperparameter optimization using Optuna (Akiba et al., 2019) with the TPE sampler. We searched over 100 configurations, with each configuration evaluated over 3 random seeds at a reduced budget of $B = 50$ oracle calls. The objective was mean HVI across seeds.

Table 2 specifies the search space. The guidance strength γ and noise scale σ were identified as the most important parameters via fANOVA importance analysis (Hutter et al., 2014).

Table 2: Hyperparameter search space for MoltenFlow.

Parameter	Type	Range	Selected
γ (guidance strength)	Log-uniform	$[10, 50]$	36.07
σ (noise scale)	Uniform	$[0.4, 0.9]$	0.80
steps (integration steps)	Integer	$[10, 50]$	12
t_{start}	Uniform	$[0.5, 0.9]$	0.89
clip_norm	Categorical	$\{, 1, 5, 10\}$	5.0
normalize_gradient	Categorical	{True, False}	True



(a) Objective contour (b) Optimization history. (c) Hyperparameter importance

Figure 8: **Hyperparameter optimization diagnostics.** Left: contour of hypervolume improvement as a function of guidance strength γ and noise scale σ. Center: optimization history across trials. Right: relative importance of hyperparameters for maximizing hypervolume improvement.

The best configuration achieved HVI $= 1.03$ (mean over 3 seeds) at trial 64, with $\gamma = 36.07$, $\sigma = 0.80$, steps $= 12$, $t_{\text{start}} = 0.89$, clip_norm $= 5.0$, and normalize_gradient $=$ True. Across all 100 trials, mean HVI was 0.72 ± 0.20, indicating substantial sensitivity to hyperparameter choice.

E.2 BUDGETED MULTI-OBJECTIVE OPTIMIZATION

E.2.1 EXPERIMENTAL DESIGN

The budgeted optimization experiment follows a similar experimental protocol for constrained oracle settings established in recent molecular optimization benchmarks (Maus et al., 2023; Tripp et al., 2020). Each method operates under a fixed budget of $B = 100$ oracle evaluations, starting from an initial set of 10 randomly sampled molecules. At each iteration, methods propose a single candidate molecule, which is decoded from the latent space and evaluated using RDKit oracles for QED and SA. The evaluated molecule is added to the candidate pool, and the process repeats until the budget is exhausted.

E.2.2 BASELINE IMPLEMENTATIONS

MoltenFlow. The full MoltenFlow pipeline uses the VAE, surrogate, and flow models described in Appendix D. At each iteration, a seed molecule is selected from the current Pareto front using diversity-weighted sampling, which balances exploitation of high-quality regions with exploration

of undersampled areas. Specifically, the selection probability for molecule i is:

$$p_i \propto w_{\text{pareto}}(i) \cdot \exp\left(-\lambda \cdot \text{sim}_{\max}(i)\right) \tag{11}$$

where $w_{\text{pareto}}(i) = 2$ if i is Pareto-optimal and 1 otherwise, $\text{sim}_{\max}(i)$ is the maximum Tanimoto similarity to recently optimized molecules, and $\lambda = 2$ is the diversity penalty.

Optimization proceeds by adding Gaussian noise ($\sigma = 0.80$) to the seed encoding and integrating the guided ODE for 12 steps starting from $t = 0.89$. Gradients are clipped to norm 5.0 and normalized before scaling by $\gamma = 36.1$.

Gradient Ascent (No-Flow Ablation). The gradient ascent baseline removes the flow velocity term from the guided dynamics:

$$\frac{dz}{dt} = \eta \cdot \nabla_z \mathcal{L}(z) \tag{12}$$

where $\eta = 0.3$ is the step size and $\mathcal{L}(z)$ is the directional objective derived from surrogate predictions. This ablation isolates the contribution of the flow prior to optimization performance. The same diversity-weighted seed selection and noise injection ($\sigma = 0.2$) are applied. These parameters were not hyperparameter optimized.

Bayesian Optimization. We implement latent-space Bayesian optimization following Gómez-Bombarelli et al. (2018) and Tripp et al. (2020). Molecules are encoded using the VAE encoder, and latent representations are aggregated via mean pooling across tokens to produce 128-dimensional vectors. For BO (MOGP), we fit a multi-output GP with a Matérn-5/2 kernel and intrinsic coregionalization (Bonilla et al., 2007). For BO (2-GP), we fit independent GPs for QED and SA. Acquisition is performed using qLogExpectedHypervolumeImprovement (Daulton et al., 2021) with 10 restarts and 512 raw samples, optimized via L-BFGS-B. We use BoTorch (Balandat et al., 2020) for GP fitting and acquisition optimization.

E.2.3 Statistical Analysis

Performance differences were assessed using the Mann-Whitney U test (Mann & Whitney, 1947), a non-parametric test appropriate for comparing distributions that may not be normally distributed. All pairwise comparisons involving MoltenFlow yielded $p < 10^{-4}$, indicating statistically significant differences. The comparison between the two BO variants was not significant ($p > 0.05$), suggesting that the multi-output GP formulation provides no advantage over independent GPs in this setting.

Confidence intervals were computed using the bootstrap percentile method with 1,000 resamples (Efron & Tibshirani, 1993). The 90% confidence level was chosen to provide tighter visual bands while maintaining statistical rigor.

E.2.4 Computational Resources

All experiments were conducted on a Nvidia A100 GPU. Per-run timings (Figure 10) reflect end-to-end optimization time including model inference, acquisition optimization (for BO), and oracle evaluation. MoltenFlow's computational advantage stems from avoiding GP fitting ($\mathcal{O}(n^3)$ in the number of evaluated points) and acquisition optimization (requiring multiple gradient-based restarts).

E.2.5 Extended Results

Figure 9 shows representative Pareto fronts from the best and worst runs for each method. MoltenFlow consistently produces dense fronts in the high-QED, low-SA region, while BO methods exhibit greater variability.

Figure 10 compares runtime distributions across methods. BO methods exhibit high variance in runtime due to the increasing cost of GP fitting as the evaluated set grows.

E.2.6 Hyperparameter Summary

Table 3 summarizes all hyperparameters for the budgeted optimization experiment.

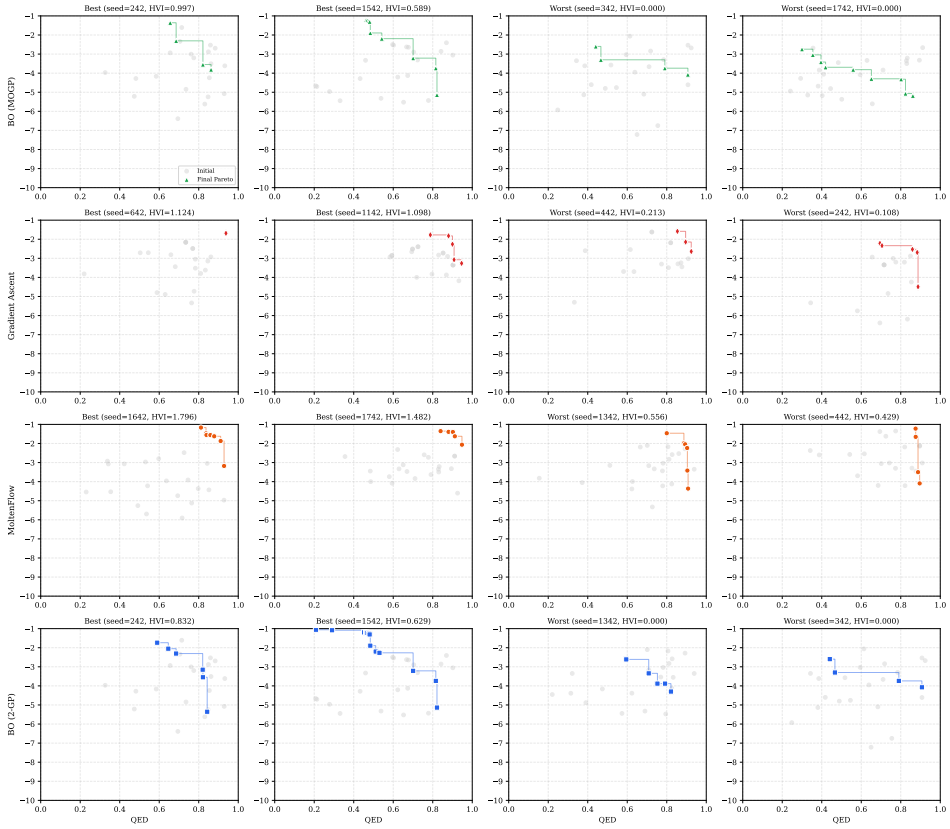


Figure 9: Representative Pareto fronts from best (top) and worst (bottom) runs for each method. Gray points: initial molecules. Colored points: final Pareto front. MoltenFlow achieves consistent improvements; BO methods show higher variance.

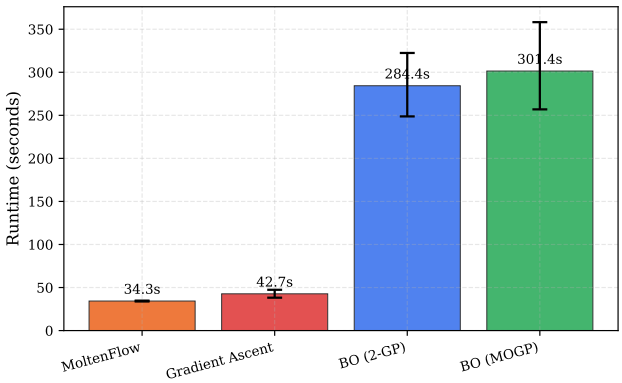


Figure 10: Runtime comparison across methods ($n = 20$ seeds). Error bars: 90% bootstrap CI. MoltenFlow is $\sim 8\times$ faster than BO methods.

E.3 Multi-Objective Pareto Front Advancement

E.3.1 Dataset and Preprocessing

We use the ZINC250K dataset (Irwin et al., 2012), containing approximately 250,000 drug-like molecules. The dataset is split into training and test sets using scaffold-based splitting to ensure chemically diverse evaluation with an 81/9/10 scaffolding split. Molecules are represented as SELF-

Table 3: Hyperparameters for budgeted optimization experiments.

Parameter	Value
Experimental Setup	
Oracle budget (B)	100
Initial set size	10
Batch size	1
Seeds per method	20
Initialization	Random sampling
MoltenFlow	
Guidance strength (γ)	36.07
Noise scale (σ)	0.80
Integration steps	12
Start time (t_{start})	0.89
Gradient clipping	5.0
Gradient normalization	True
Seed selection	Diversity-weighted
Diversity penalty (λ)	2.0
Pareto weight	2.0
Gradient Ascent	
Step size (η)	0.3
Noise scale (σ)	0.2
Steps	10
Seed selection	Diversity-weighted
Bayesian Optimization	
GP kernel	Matérn-5/2
Latent aggregation	Mean pooling
Acquisition restarts	10
Raw samples	512
Hyperparameter Optimization	
HPO method	TPE (Optuna)
Trials	100
Seeds per trial	3
Budget per trial	50
Evaluation	
Confidence level	0.90
Bootstrap samples	1,000
Reference point	$(0.0, -10.0)$

IES strings (Krenn et al., 2020), which guarantee syntactic validity upon decoding. The vocabulary contains 111 tokens with a maximum sequence length of 128.

E.3.2 Molecular Properties

We compute three molecular properties using RDKit (Landrum):

Quantitative Estimate of Drug-likeness (QED). QED (Bickerton et al., 2012) is a composite score that quantifies drug-likeness based on eight molecular descriptors: molecular weight, octanol-water partition coefficient (ALogP), number of hydrogen bond donors, number of hydrogen bond acceptors, molecular polar surface area, number of rotatable bonds, number of aromatic rings, and number of structural alerts. Each descriptor is mapped to a desirability function based on the distributions observed in known drugs, and the final QED score is the geometric mean of all desirability values:

$$\text{QED} = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\ln d_i\right) \tag{13}$$

where d_i is the desirability function for descriptor i. QED $\in [0, 1]$, with higher values indicating more drug-like molecules.

Synthetic Accessibility Score (SA). SA (Ertl & Schuffenhauer, 2009) estimates the ease of chemical synthesis based on fragment contributions and molecular complexity:

$$\text{SA} = \text{fragmentScore} - \text{complexityPenalty} \tag{14}$$

where the fragment score is derived from the frequency of molecular fragments in the PubChem database (common fragments receive higher scores), and the complexity penalty accounts for features that complicate synthesis such as macrocycles, stereocenters, and spiro/bridged systems. SA $\in [1, 10]$, with lower values indicating easier synthesis.

Penalized LogP (pLogP). pLogP (Jin et al., 2019; You et al., 2019) combines lipophilicity with synthesizability and ring size constraints:

$$\text{pLogP} = \log P - \text{SA} - \text{ringPenalty} \tag{15}$$

where $\log P$ is the Crippen octanol-water partition coefficient, and $\text{ringPenalty} = \max(0, \text{largestRingSize} - 6)$ penalizes molecules with unusually large rings. This metric is commonly used in molecular optimization benchmarks as it captures a trade-off between desirable lipophilicity and practical constraints.

E.3.3 Optimization Procedure

Optimization initializes from latent encodings of test-set Pareto-optimal molecules (identified using surrogate predictions). Gaussian noise ($\sigma = 0.8$) is added to each latent code. Guided integration uses 12 Euler steps from $t_{\text{start}} = 0.89$ to $t = 1$ (per the parameters obtained in E.1) restricting the flow to the final 11% of its trajectory where representations are near the data manifold. The directional objective is derived from normalized gradients pointing toward improved Pareto regions. Candidate molecules for optimization are selected from the test set ($n = 1000$ candidates).

E.3.4 Evaluation Metrics

We evaluate generated and optimized molecules using a comprehensive set of metrics:

Basic set metrics.

- **Validity:** Fraction of generated molecules that are parseable by RDKit.
- **Uniqueness:** Fraction of unique canonical SMILES among valid generated molecules.
- **Novelty:** Fraction of unique valid molecules not present in the training set.

Hypervolume metrics.

- **Hypervolume (HV):** Volume of objective space dominated by the Pareto front, bounded by a reference point. Computed using the pymoo library (Blank & Deb, 2020).
- **Hypervolume improvement (HVI):** $\text{HVI} = \text{HV}(\text{baseline} \cup \text{optimized}) - \text{HV}(\text{baseline})$.
- **Reference point:** Automatically selected with 10% margin beyond worst observed values: $(0.033, 7.90)$ for (QED, SA).
- **Bootstrap confidence intervals:** 95% CIs computed over 1,000 bootstrap resamples of the Pareto front.

Surrogate fidelity.

- **MSE:** Mean squared error between surrogate predictions and oracle (RDKit-computed) values on optimized molecules.
- **R^2:** Coefficient of determination. Negative values indicate the surrogate predicts worse than a constant baseline, signaling over-optimization beyond the surrogate's reliable regime.

Distribution metrics.

- **Fréchet distance (FD-FP):** Fréchet distance computed on Morgan fingerprint embeddings (radius 2, 2048 bits) projected to 256 dimensions via random Gaussian projection. Measures distributional shift from the training set.
- **Descriptor KL divergences:** KL divergence $D_{\text{KL}}(P_{\text{gen}} \| P_{\text{train}})$ computed for seven molecular descriptors using 50-bin histograms:
  - **MolWt:** Molecular weight (Daltons)
  - **MolLogP:** Crippen octanol-water partition coefficient
  - **HBD:** Number of hydrogen bond donors
  - **HBA:** Number of hydrogen bond acceptors
  - **TPSA:** Topological polar surface area (Å^2)
  - **RingCount:** Number of rings
  - **RotBonds:** Number of rotatable bonds
- **Average descriptor KL:** Mean KL divergence across all seven descriptors.

Scaffold metrics.

- **Scaffold diversity:** Fraction of unique Bemis-Murcko scaffolds (Bemis & Murcko, 1996) among valid molecules. Values near 1.0 indicate high structural diversity; low values indicate scaffold collapse.

E.4 Gamma Sweep Experimental Details

This section provides additional details on the guidance strength (γ) characterization experiment described in Section 3.1.2.

E.4.1 Two-Phase Search Strategy

We employ a two-phase search to efficiently identify the optimal guidance strength without exhaustive enumeration.

Phase 1: Coarse Sweep. The first phase evaluates gamma values spanning five orders of magnitude on a logarithmic scale:

$$\gamma \in \{0.001, 0.01, 0.1, 1.0, 10, 100, 1000\} \tag{16}$$

Each configuration is evaluated with three random seeds (42, 123, 456) to quantify variance. This coarse sweep identifies $\gamma = 100$ as the best-performing region.

Phase 2: Bayesian Optimization. The second phase refines the search around the best coarse region using tree-structured Parzen estimators (TPE). The search range is set to $\pm 3\times$ the best coarse gamma:

$$\gamma \in [33.3, 300.0] \tag{17}$$

We run 30 Optuna trials, each evaluated with a single seed. The objective is to maximize hypervolume improvement (HVI). This phase identifies $\gamma^* = 92.4$ as optimal, achieving HVI $= 0.405$.

E.4.2 Optimization Hyperparameters

We use the same hyperparameters outlined in E.1 with the exception of varying the guidance γ.

E.4.3 Full Results

Table 4 presents the complete results across all evaluated gamma values, including hypervolume improvement, surrogate model fidelity, and distributional metrics. Table 5 provides the per-descriptor KL divergences measuring distributional shift from the training data.

Table 4: Complete gamma sweep results. HVI: hypervolume improvement. CI: mean ± std across 3 seeds. Unique/Novel: fraction among valid molecules. All configurations achieve 100% validity except $\gamma = 1000$ (unique: 13%).

γ	HVI↑	HVI (%)↑	HVI CI↑	MSE↓	Unique (%)↑	Novel (%)↑	Scaf. Div.↑	FD-FP↓	Avg. KL↓
0.001	0.023	0.39	0.023 ± 0.016	0.63	100	100	0.962	8.89	0.18
0.1	0.023	0.39	0.023 ± 0.016	0.64	100	100	0.963	8.91	0.18
1	0.023	0.39	0.023 ± 0.016	0.65	100	100	0.961	8.92	0.18
10	0.019	0.33	0.019 ± 0.015	0.77	100	100	0.956	9.00	0.18
$\mathbf{92.4}^*$	**0.405**	**6.96**	—	1.98	100	100	0.743	13.18	0.47
100	0.357	6.13	0.357 ± 0.077	2.02	100	100	0.702	13.46	0.53
1000	0.170	2.91	0.170 ± 0.003	4.52	13	100	0.000	40.81	8.99

Table 5: Individual descriptor KL divergences $D_{\text{KL}}(P_{\text{gen}} \| P_{\text{train}})\downarrow$ across guidance strengths.

γ	MolWt	MolLogP	HBD	HBA	TPSA	RingCount	RotBonds
0.001	0.22	0.13	0.02	0.16	0.14	0.12	0.48
0.1	0.22	0.12	0.02	0.16	0.14	0.12	0.46
1	0.22	0.13	0.02	0.15	0.15	0.12	0.46
10	0.24	0.13	0.02	0.17	0.14	0.11	0.42
$\mathbf{92.4}^*$	0.68	0.22	0.03	0.76	0.67	0.27	0.68
100	0.76	0.23	0.05	0.86	0.77	0.32	0.71
1000	16.91	15.54	0.67	3.36	5.04	5.19	16.20

E.4.4 Interpretation of Results

The gamma sweep reveals three distinct operating regimes:

Conservative regime ($\gamma \leq 10$). Guidance is too weak to overcome the flow prior. Molecules remain near their original positions in latent space, resulting in minimal property changes (HVI $< 0.4\%$) but excellent scaffold preservation (diversity > 0.95) and low distributional shift (Fréchet < 9.1).

Optimal regime ($\gamma \approx 50$–150). Guidance is strong enough to push molecules toward improved property regions while the surrogate model remains predictive. The optimal $\gamma^* = 92.4$ achieves maximum HVI (6.96%) with acceptable trade-offs: scaffold diversity remains at 74%, and Fréchet distance increases moderately to 13.2.

Over-optimization regime ($\gamma > 200$). Guidance dominates the flow prior, causing optimization trajectories to escape the surrogate's reliable predictive regime. Symptoms include: (1) decreasing HVI despite stronger guidance, (2) collapsing scaffold diversity (0% at $\gamma = 1000$), (3) severe surrogate failure ($R^2 = -40.2$), and (4) large distributional shift (Fréchet $= 40.8$).

The non-monotonic relationship between γ and HVI—rising until $\gamma \approx 100$ then declining—is the signature of surrogate exploitation. At extreme γ, the optimizer finds molecules that score well according to the surrogate but poorly according to the oracle, having left the manifold where the surrogate was trained.

E.4.5 HYPERPARAMETER SUMMARY

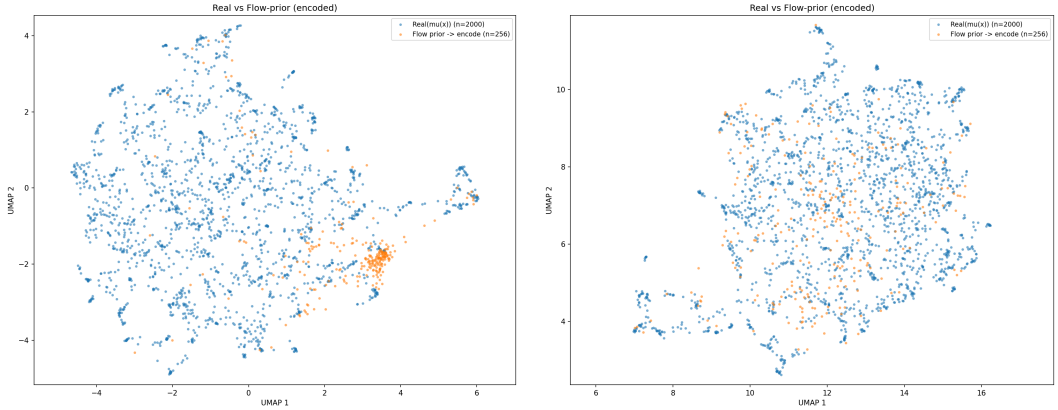
Table 6 summarizes key hyperparameters.

Table 6: Hyperparameters for ZINC optimization experiments.

Parameter	**Value**
Data	
Dataset	ZINC250K
Representation	SELFIES
Max sequence length	128
Vocabulary size	111
Train/test split	81/9/10
VAE (Pretrain)	
Encoder/decoder layers	6
Hidden dimension (d_{model})	128
Attention heads	8
Feedforward dimension	1024
Latent dimension	128
Codebook size (K)	8
Dropout	0.1
Pretrain epochs	150
Batch size	256
Learning rate	10^{-4}
KL weight ($\beta_{\max}$)	0.1
KL warmup fraction	0.35
VAE (Finetune)	
Finetune epochs	20
Batch size	1024
Learning rate	10^{-3}
Property weight	1.0
Surrogate	
Hidden dimension	1024
Aggregation	Mean pooling
Dropout	0.1
Flow model	
Transformer layers	10
Hidden dimension (d_{model})	256
Attention heads	8
Feedforward dimension	512
Time embedding dimension	128
Dropout	0.1
Training epochs	100
Batch size	1024
Learning rate	2×10^{-4}
Optimization	
Candidates (n)	1000
Noise scale (σ)	0.80
Integration start (t_{start})	0.89
Integration steps	12
Evaluation	
Reference point margin	0.1
Bootstrap samples	1000
Confidence level	0.95

E.5 MOLECULAR GENERATION AND ABLATIONS

Additional ablation results and qualitative analyses of MoltenFlow's molecular generation behavior, complementing the optimization-focused experiments in the main text.



(a) SMILES model: real vs flow prior (re-encoded). (b) SELFIES model: real vs flow prior (re-encoded).

Figure 11: UMAP of posterior-mean latent embeddings $\mu_\phi(x)$ for real molecules (fit set) and generated molecules after re-encoding. Each UMAP is fit on the real embeddings for the corresponding model run.