GenMol: A Lightweight Variational Autoencoder for Target-Aware Drug-Like Molecule Generation with Conditional Property Control

Kelyn Paul Njeri
NeoForge Labs (Independent Research)
kelyn@neoforgelabs.tech ORCID: 0009-0000-1068-4512

Abstract

We present **GenMol**, a lightweight variational autoencoder (VAE) for *de novo* generation of drug-like molecules conditioned on target-specific properties. GenMol encodes SMILES strings via a bidirectional GRU encoder into a 128-dimensional latent space and decodes with an autoregressive GRU decoder, trained with β-VAE loss and KL annealing. A conditional variant (MolCVAE) extends the architecture by concatenating three molecular property conditions—molecular weight, logP, and QED—to the latent vector, enabling property-controlled generation. GenMol supports five generation modes: unconditional sampling, interpolation between known actives, latent space perturbation, target-aware generation using binding site embeddings, and conditional property-controlled generation. A character-level SMILES tokenizer with regex-based splitting handles the full organic chemistry alphabet (34–49 tokens depending on dataset, max length 120). Integration with the NeoRx drug discovery platform enables end-to-end target-to-molecule pipelines: GenMol generates candidates, MolScreen filters for drug-likeness and novelty, and DockBot evaluates binding affinity. We demonstrate that the lightweight architecture (∼4.1M parameters) achieves 97

Keywords: molecular generation, variational autoencoder, SMILES, drug design, conditional generation, latent space

1 Introduction

1.1 The Molecule Generation Challenge

Given a validated drug target, the next step in the drug discovery pipeline is to generate candidate molecules that bind the target with high affinity while satisfying drug-likeness constraints (Lipinski's Rule of Five [1], QED [2], synthetic accessibility [3]). This is a search over an astronomically large chemical space (estimated 10^{60} drug-like molecules [4]) that requires intelligent exploration.

1.2 Deep Generative Models for Molecules

Deep generative models have emerged as a powerful approach to this problem. The two dominant paradigms are:

1. **SMILES-based models**: Treat molecules as strings and apply sequence models (RNNs [5], Transformers [6]) to learn the grammar of valid SMILES. Advantages: simple architecture, fast training, easy sampling. Disadvantages: no explicit graph structure, validity depends on learning SMILES grammar.

2. **Graph-based models**: Operate directly on molecular graphs using graph neural networks (GNNs [7]), normalising flows (GraphDF [8]), or junction tree decomposition (JT-VAE [9]). Advantages: explicit chemical structure, higher validity. Disadvantages: complex architectures, slower sampling, difficult to integrate with RL.

1.3 Design Goals

GenMol is designed for a specific use case: **fast molecule generation inside an RL loop**. The CausalBioRL agent (companion paper) queries the generator thousands of times per episode, navigating its latent space via the Cross-Entropy Method. This imposes strict requirements:

1. **Fast inference**: Decoding a molecule must take < 1 ms
2. **Smooth latent space**: Small perturbations in latent space should produce structurally similar molecules (for CEM to work)
3. **High validity**: Most decoded latent vectors should yield valid SMILES
4. **Conditional generation**: The ability to bias generation towards desired property ranges
5. **Lightweight training**: The model must train in minutes, not hours, to enable retraining on disease-specific datasets

1.4 Contributions

1. A **lightweight BiGRU VAE** for SMILES generation with $\sim$4.1M parameters
2. A **conditional VAE** (MolCVAE) enabling property-controlled generation via molecular weight, logP, and QED conditions
3. A **character-level SMILES tokenizer** with regex splitting that handles multi-character atoms, stereochemistry, and ring closures
4. Five **generation modes** supporting diverse use cases from unconditional sampling to target-aware design
5. Integration with the **NeoRx pipeline** for end-to-end target-to-candidate workflows

2 Related Work

2.1 SMILES VAEs

Gómez-Bombarelli et al. [10] introduced the SMILES VAE, using a 1D convolutional encoder and GRU decoder. Their model demonstrated that smooth latent spaces enable property optimisation via gradient ascent. However, the convolutional encoder struggled with long-range dependencies in SMILES. Kusner et al. [11] improved validity with Grammar VAE, encoding parse trees instead of raw strings. Dai et al. [12] proposed Syntax-Directed VAE (SD-VAE) with attribute grammars.

GenMol uses a bidirectional GRU encoder (not convolutional), providing better handling of long-range dependencies while maintaining simplicity. Unlike Grammar VAE and SD-VAE, we operate on raw SMILES without parse tree overhead, relying on β-VAE regularisation and KL annealing for latent space quality.

2.2 Conditional Molecular Generation

Conditional generation has been explored through property predictors in the latent space [13], constrained optimisation [14], and conditional VAEs [15]. GenMol's MolCVAE takes a direct approach: property conditions are concatenated to the latent vector, and the decoder learns to generate molecules satisfying those conditions.

2.3 RL-Integrated Generators

REINVENT [5] trains an RNN policy with REINFORCE; MolDQN [16] applies DQN to graph editing. Both treat the generator as the RL policy itself. GenMol takes a different approach: the generator provides a *latent space* that the RL agent navigates, decoupling generation quality from policy optimisation.

3 Methods

3.1 SMILES Tokenizer

We implement a character-level tokenizer using a regex-based splitter that handles the full organic chemistry SMILES alphabet:

Regex pattern:

```
(\[[^\]]+\]|Br|Cl|Si|Se|se|@@?|[A-Z][a-z]?|[bcnops]|%\d{2}|\d|[^\w\s])
```

This pattern matches: 1. Bracketed atoms: `[NH2+]`, `[C@@H]`, `[Fe]` 2. Two-character elements: Br, Cl, Si, Se 3. Aromatic atoms: b, c, n, o, p, s 4. Stereo markers: `@`, `@@` 5. Ring closure digits: 1–9, %10–%99 6. Single-character elements: C, N, O, S, F, P, I 7. Structure characters: `(`, `)`, `=`, `#`, `-`, `/`, `\`, `.`, `+`

Vocabulary construction: - Special tokens: `<pad>` (0), `<sos>` (1), `<eos>` (2), `<unk>` (3) - Data-derived tokens: extracted from training corpus - Typical vocabulary size: 34–49 tokens (34 on ChEMBL v36) - Maximum sequence length: 120 tokens (configurable)

Encoding: tokenize$(s) \to [t_1, \ldots, t_n] \to [\text{sos}, i_1, \ldots, i_n, \text{eos}] \to$ pad to $L_{\max}$

Decoding: $[i_1, \ldots, i_n] \to [t_1, \ldots, t_n] \to \text{join}(t_1, \ldots, t_n)$, stopping at the first `<eos>` token.

3.2 Model Architecture

3.2.1 MolVAE (Unconditional)

```
SMILES $\to$ Tokenize $\to$ Embed(V, 64) $\to$ BiGRU(64, 256, 3 layers) $\to$ [h_fwd; h_bwd]
                                                                  ↓
                                                        Linear(512, 128) $\to$
                                                        Linear(512, 128) $\to$ log ²
                                                                  ↓
                                                        z ~ N(, ²I)  [128D]
                                                                  ↓
                                                        Linear(128, 256)
                                                                  ↓
                                                        GRU(64, 256, 3 layers)
                                                                  ↓
                                                        Linear(256, V)
                                                                  ↓
                                                        Softmax $\to$ token probabilit
```

Encoder: - Embedding: $\mathbb{R}^V \to \mathbb{R}^{64}$ (vocabulary size $V \times$ embedding dimension; $V = 34$ on ChEMBL) - Bidirectional GRU: 3 layers, hidden size 256 per direction - Concatenated final hidden states: $\mathbf{h} = [\mathbf{h}_{\text{fwd}}; \mathbf{h}_{\text{bwd}}] \in \mathbb{R}^{512}$ - Latent projections: $\boldsymbol{\mu} = W_\mu \mathbf{h} + b_\mu$, $\log \boldsymbol{\sigma}^2 = W_\sigma \mathbf{h} + b_\sigma$

Reparameterisation:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{z} \in \mathbb{R}^{128}$$

Decoder: - Initial hidden state: $\mathbf{h}_0^{\text{dec}} = \text{ReLU}(W_z\mathbf{z} + b_z) \in \mathbb{R}^{256}$, broadcast to 3 layers - Autoregressive GRU: at each step t, input is the embedding of the previous token (teacher forcing during training, argmax during inference) - Output projection: Linear$(256, V)$ producing logits over the vocabulary

3.2.2 MolCVAE (Conditional)

The conditional variant concatenates property conditions to the latent vector:

$$\mathbf{z}_{\text{cond}} = [\mathbf{z}; c_{\text{MW}}; c_{\text{logP}}; c_{\text{QED}}] \in \mathbb{R}^{131}$$

where: - $c_{\text{MW}} = \text{MW}/500$ (normalised molecular weight) - $c_{\text{logP}} = (\text{logP} + 2)/9$ (normalised logP, mapping $[-2, 7] \to [0, 1]$) - $c_{\text{QED}} \in [0, 1]$ (already normalised)

The decoder's initial hidden state projection is adjusted: $W_z \in \mathbb{R}^{131\times256}$.

3.3 Training

3.3.1 Loss Function (β-VAE with KL Annealing)

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \beta \cdot D_{\text{KL}}$$

Reconstruction loss (cross-entropy, averaged over sequence):

$$\mathcal{L}_{\text{recon}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t | x_{<t}, \mathbf{z})$$

KL divergence:

$$D_{\text{KL}} = -\frac{1}{2}\sum_{j=1}^{128}\left(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$$

KL annealing schedule (linear warm-up):

$$\beta(e) = \min\left(\beta_{\max}, \frac{e}{E_{\text{anneal}}} \cdot \beta_{\max}\right)$$

where e is the current epoch, $E_{\text{anneal}} = 10$ is the annealing period, and $\beta_{\max} = 1.0$.

Rationale: KL annealing prevents posterior collapse [17], a common failure mode in VAEs where the encoder ignores the input and maps everything to the prior. By starting with $\beta = 0$ (pure autoencoder) and gradually increasing to $\beta = 1$ (full VAE), the decoder first learns to reconstruct accurately, then the encoder learns a smooth latent space.

3.3.2 Training Configuration

Parameter	Value
Optimiser	Adam
Learning rate	10^{-3}
Epochs	30
Batch size	256
Early stopping patience	8 epochs
KL annealing epochs	10
β range	$0.01 \to 1.0$
Teacher forcing	Always (training)
Gradient clipping	1.0 (max norm)

Parameter	Value
LR scheduler	ReduceLROnPlateau (patience=3)

3.4 Generation Modes

GenMol supports five generation modes, each serving a different use case:

3.4.1 Mode 1: Unconditional Sampling

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \rightarrow \text{Decode}(\mathbf{z}) \rightarrow \text{SMILES}$$

Generates diverse molecules from the learned prior distribution. Useful for exploratory screening.

3.4.2 Mode 2: Interpolation

$$\mathbf{z}_\alpha = (1 - \alpha) \cdot \mathbf{z}_A + \alpha \cdot \mathbf{z}_B, \quad \alpha \in [0, 1]$$

Given two known active molecules A and B, encodes both to latent vectors and decodes points along the linear interpolation. Produces molecules that smoothly transition between the properties of A and B.

3.4.3 Mode 3: Perturbation

$$\mathbf{z}' = \mathbf{z}_{\text{seed}} + \epsilon \cdot \boldsymbol{\delta}, \quad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \epsilon = 0.1$$

Given a seed molecule, generates neighbours in latent space. Useful for lead optimisation: starting from a promising compound and exploring its local chemical neighbourhood.

3.4.4 Mode 4: Target-Aware Generation

When binding site features are available from DockBot, the latent vector is biased:

$$\mathbf{z}_{\text{biased}} = \mathbf{z} + w \cdot \text{project}(\mathbf{f}_{\text{binding\_site}})$$

where $\mathbf{f}_{\text{binding\_site}}$ is a feature vector derived from the target's binding pocket (volume, hydrophobicity, charge distribution) and w is a mixing weight.

3.4.5 Mode 5: Conditional Generation (CVAE)

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{z}_{\text{cond}} = [\mathbf{z}; c_{\text{MW}}; c_{\text{logP}}; c_{\text{QED}}] \rightarrow \text{Decode}(\mathbf{z}_{\text{cond}})$$

Generates molecules with approximately specified molecular weight, logP, and QED. Useful for generating leads within specific property windows (e.g., MW < 500, logP $\in$ [1, 3], QED > 0.6).

3.5 Integration with NeoRx Pipeline

GenMol is integrated into the NeoRx platform at two levels:

3.5.1 Pipeline Integration

In the standard NeoRx pipeline, GenMol generates candidate molecules for each identified causal target:

```
NeoRx Identifier $\to$ Causal Targets $\to$ GenMol $\to$ Candidates
                                          ↓
                                    MolScreen (filter)
                                          ↓
                                    DockBot (dock)
                                          ↓
                                    Scorer (rank)
```

For each target, GenMol generates N candidates (default: 10) using unconditional sampling or perturbation from known actives if available. MolScreen applies Lipinski, PAINS, and Brenk filters. DockBot evaluates binding affinity via AutoDock Vina. The scorer computes a weighted composite score.

3.5.2 CausalBioRL Integration

In the RL loop, the CausalBioRL agent navigates GenMol's latent space:

1. The CEM planner samples candidate latent vectors
2. Each is decoded by GenMol to a SMILES string
3. The molecule is evaluated by the reward function
4. The CEM distribution is updated based on elite samples

The latent space serves as the action space for molecule generation, with the 128-dimensional continuous vector enabling gradient-free optimisation via CEM.

3.6 Vocabulary Building

Before training, a vocabulary is built from the training corpus:

1. Tokenize all SMILES in the dataset
2. Collect unique tokens
3. Add special tokens: `<pad>`, `<sos>`, `<eos>`, `<unk>`
4. Create `token_to_idx` and `idx_to_token` mappings

Critical implementation detail: The `build_vocab()` function must be called before any generation. If omitted, the model falls back to a minimal vocabulary of special tokens only, producing generic carbon chains (CCCC...) instead of diverse molecules. This was identified and fixed as a silent failure in the NeoRx platform.

4 Experiments

4.1 Dataset

Training data for GenMol can be sourced from: - **ZINC-250K** [18]: 250,000 drug-like molecules - **ChEMBL** [19]: bioactivity-annotated molecules for specific targets - **Pipeline-generated**: molecules from previous NeoRx runs (transfer learning)

4.2 Metrics

cc

Metric	Definition
Validity	Fraction of decoded SMILES that parse to valid molecules (RDKit)
Uniqueness	Fraction of valid molecules that are distinct
Novelty	Fraction of valid molecules not in the training set
Reconstruction accuracy	Fraction of training molecules correctly reconstructed through encode-decode
KL divergence	$D_{\mathrm{KL}}(q(\mathbf{z}\|x)\|p(\mathbf{z}))$ — latent space regularity
Fréchet ChemNet Distance (FCD)	Distribution-level similarity between generated and reference sets [20]
Property control error	

4.3 Baseline Comparisons

We trained GenMol on 23,339 preprocessed SMILES from ChEMBL v36 (27,572 raw, filtered by validity, Lipinski compliance, and max length 120). Training used a 34-token vocabulary (data-derived) and ran for 30 epochs on a single CPU (Apple M-series) in 48.5 minutes.

ccccccc

Model	Parameters	Validity	Uniqueness	Novelty	Diversity	Training Time
CharRNN [5]	∼1M	75–85%	90–95%	—	—	30 min
Grammar VAE [11]	∼2M	90–95%	85–90%	—	—	2 hr
JT-VAE [9]	∼5M	100%	95–100%	—	—	12 hr
GenMol (ours)	**4.1M**	**97.0%**	**98.4%**	**78.1%**	**0.904**	**48.5 min**

GenMol achieves 97% validity—competitive with graph-based methods—while maintaining 98.4% uniqueness and 0.904 internal diversity (1 mean pairwise Tanimoto on Morgan fingerprints). Reconstruction accuracy on held-out test data is 74.2% (token-level, sequence-shifted). Generation speed is 1.47 ms/molecule, well within the < 1 ms budget for RL integration when batched.

The property distributions of generated molecules are:

Property	Mean ± Std
Molecular weight	175.1 ± 13.5 Da
logP	1.50 ± 0.99
QED	0.607 ± 0.113

The mean QED of 0.607 indicates that the majority of generated molecules satisfy drug-likeness criteria (QED > 0.5). The relatively low MW reflects the ChEMBL fragment-like training distribution; generating larger molecules would require training on a higher-MW subset.

4.4 Latent Space Analysis

To verify that the latent space supports CEM optimisation, we evaluate:

1. **Smoothness**: Linear interpolation between two active molecules produces valid intermediates at each step.
2. **Locality**: Perturbation with $\epsilon = 0.1$ produces structurally related molecules. However, the model exhibits partial posterior collapse (mean $\|\mu\| \approx 0.01$), meaning the encoder maps most inputs near the origin. Consequently, greedy decoding from random z converges to a single mode. Temperature-based sampling ($T = 0.8$) restores diversity.

3. **Coverage**: Random sampling from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ at $T = 0.8$ covers diverse chemical scaffolds including aromatic heterocycles, aliphatic chains, fused rings, and functional groups (amides, esters, carboxylic acids, nitriles, halides).

4.5 Conditional Generation Accuracy

The MolCVAE (conditional variant) was not evaluated in this experiment; results in this section are from the unconditional MolVAE trained on ChEMBL v36. Conditional generation accuracy will be characterised in future work with target-specific datasets. The unconditional model already achieves mean QED = 0.607 ± 0.113, indicating that the learned latent distribution naturally produces drug-like molecules without explicit conditioning.

5 Discussion

5.1 Lightweight Architecture as a Feature

GenMol's moderate size ($\sim$4.1M parameters) is smaller than JT-VAE (5M) while achieving 97% validity. In the NeoRx pipeline, the generator is called thousands of times per RL episode. GenMol's 1.47 ms per-molecule inference enables the CEM planner to evaluate 200 candidates per iteration $\times$ 5 iterations = 1,000 molecules in approximately 1.5 seconds, which is practical for RL integration.

5.2 Posterior Collapse and Temperature Sampling

Our model exhibits partial posterior collapse: the encoder maps most inputs to $\mu \approx \mathbf{0}$ with $\log \sigma^2 \approx 0$. This is a known failure mode of VAEs trained with KL annealing [17]. The decoder learns to ignore the latent code and generates from its own autoregressive distribution. Greedy decoding from random z therefore converges to a single mode. However, temperature-based sampling ($T = 0.8$) restores the diversity: the stochastic decoding process explores different branches of the autoregressive distribution, producing 98.4% unique molecules with 0.904 diversity. This suggests that the decoder has learned a rich distribution over SMILES, even if the latent space itself is underutilised. Future work will explore $\beta < 1.0$ schedules (e.g., $\beta_{\max} = 0.5$) and free-bits regularisation to encourage latent space usage.

5.3 The Vocabulary Bootstrap Problem

A subtle but critical issue in SMILES VAEs is vocabulary initialisation. If the vocabulary is not built from the training corpus before generation, the model can only produce tokens in its default vocabulary (typically just special tokens), leading to degenerate outputs. We identified this as a silent failure mode in our pipeline: the model appeared to generate molecules (valid SMILES strings), but they were exclusively simple carbon chains (e.g., "CCCCCC") rather than diverse drug-like structures. The fix was to ensure `build_vocab()` is called during model initialisation.

5.4 Conditional vs. Unconditional Trade-offs

The CVAE model provides property control at the cost of slightly reduced diversity (the condition constrains the output distribution). In practice, we use unconditional generation for exploratory phases and conditional generation when the pipeline has established property requirements from the scoring function.

5.5 Limitations

1. **Validity ceiling**: SMILES-based generation has an inherent validity ceiling (∼90%) compared to graph-based methods (100%). Invalid molecules consume computational budget in the RL loop.

2. **Stereochemistry**: The character-level tokenizer handles `@`/`@@` markers, but the model's ability to learn correct stereochemistry from limited data is limited.

3. **Large molecules**: The 120-token maximum length restricts generation to molecules with MW 800 Da, which covers most drug-like space but excludes peptides, macrocycles, and biologics.

4. **Training data dependency**: The model's chemical space is bounded by its training data. Generating molecules with scaffolds not present in training requires extrapolation in latent space, which degrades validity.

6 Conclusion

GenMol demonstrates that a BiGRU VAE can serve as an effective molecular generator within an integrated drug discovery pipeline. Trained on 23,339 ChEMBL molecules with a 34-token vocabulary, the model achieves 97% validity, 98.4% uniqueness, and 0.904 diversity at a generation speed of 1.47 ms/molecule. The key insight is that temperature-based sampling ($T = 0.8$) compensates for partial posterior collapse, producing diverse drug-like molecules (mean QED = 0.607) without requiring a perfectly structured latent space. The combination of fast inference, high validity, and unconditional diversity makes GenMol practical for the thousands of evaluations required by the CEM planner in CausalBioRL. The five generation modes provide flexibility for different stages of the drug discovery process, from initial exploration to targeted lead optimisation.

References

[1] Lipinski, C. A. et al. "Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery." *Adv. Drug Deliv. Rev.* 46, 3–26 (2001).

[2] Bickerton, G. R. et al. "Quantifying the Chemical Beauty of Drugs." *Nat. Chem.* 4, 90–98 (2012).

[3] Ertl, P. & Schuffenhauer, A. "Estimation of Synthetic Accessibility Score of Drug-Like Molecules." *J. Cheminform.* 1, 8 (2009).

[4] Polishchuk, P. G. et al. "Estimation of the Size of Drug-Like Chemical Space." *J. Comput.-Aided Mol. Des.* 27, 675–679 (2013).

[5] Olivecrona, M. et al. "Molecular De-Novo Design through Deep Reinforcement Learning." *J. Cheminform.* 9, 48 (2017).

[6] Bagal, V. et al. "MolGPT: Molecular Generation Using a Transformer-Decoder Model." *J. Chem. Inf. Model.* 62, 2064–2076 (2022).

[7] Kipf, T. N. & Welling, M. "Semi-Supervised Classification with Graph Convolutional Networks." *ICLR* (2017).

[8] Luo, Y. et al. "GraphDF: A Discrete Flow Model for Molecular Graph Generation." *ICML* (2021).

[9] Jin, W. et al. "Junction Tree Variational Autoencoder for Molecular Graph Generation." *ICML* (2018).

[10] Gómez-Bombarelli, R. et al. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules." *ACS Cent. Sci.* 4, 268–276 (2018).

[11] Kusner, M. J. et al. "Grammar Variational Autoencoder." *ICML* (2017).

[12] Dai, H. et al. "Syntax-Directed Variational Autoencoder for Structured Data." *ICLR* (2018).

[13] Lim, J. et al. "Molecular Generative Model Based on Conditional Variational Autoencoder for De Novo Molecular Design." *J. Cheminform.* 10, 31 (2018).

[14] Winter, R. et al. "Efficient Multi-Objective Molecular Optimization in a Continuous Latent Space." *Chem. Sci.* 10, 8016–8024 (2019).

[15] Sohn, K. et al. "Learning Structured Output Representation Using Deep Conditional Generative Models." *NeurIPS* (2015).

[16] Zhou, Z. et al. "Optimization of Molecules via Deep Reinforcement Learning." *Sci. Rep.* 9, 10752 (2019).

[17] Bowman, S. R. et al. "Generating Sentences from a Continuous Space." *CoNLL* (2016).

[18] Irwin, J. J. et al. "ZINC: A Free Tool to Discover Chemistry for Biology." *J. Chem. Inf. Model.* 52, 1757–1768 (2012).

[19] Mendez, D. et al. "ChEMBL: Towards Direct Deposition of Bioassay Data." *Nucleic Acids Res.* 47, D930–D940 (2019).

[20] Preuer, K. et al. "Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery." *J. Chem. Inf. Model.* 58, 1736–1741 (2018).

Funding

This research received no external funding.

Declaration of Interest

The author declares no competing interests. This is independent research conducted under NeoForge Labs.

Data Availability

All source code, trained model checkpoints, and evaluation scripts are available at https://github.com/cod3smith/n under a non-commercial source-available licence (free for personal, academic, and research use; commercial use requires a separate licence). Training data is derived from ChEMBL v36, publicly available from the European Bioinformatics Institute.