MoFlowGAN: Combining adversarial and likelihood learning for *de novo* molecular generation

Nathan M. Lui[*] **Max Li** **Matthew Ford**
Cornell University, Ithaca, NY, USA
`{nml64, mdl262, mtf62}@cornell.edu`

Abstract

Deep generative models for molecular graphs offer a new avenue for property optimization in drug discovery. Optimizing differentiable models that generate molecular graphs is certainly faster, cheaper, and much more accessible than traditional methods of chemical synthesis. Recent advances in generative modeling have managed address many of the challenges surrounding generation of chemically-valid molecular graphs from latent representations, however the question of generating high-quality molecules remains. Herein we introduce MoFlowGAN a tandem normalizing flow model that can be trained on both adversarial and reward objectives. We train our model on QM9 to generate high quality and *drug-like* compounds. Our experiments show that MoFlowGAN is competitive with current state-of-the-art generative models while requiring far fewer training resources.

1 Introduction

Traditional methods of drug discovery and molecular design rely on the resource-intensive process of using human expertise to propose, synthesize, and test novel compounds. This approach, while historically fruitful, is slow, expensive, and yields molecules with a striking lack of diversity [1]. Recently, deep generative models have shown their ability to streamline the discovery process [2–4], inverting the traditional drug design problem; given a set of desired properties, what are the molecules that satisfy our requirements? By learning a function that maps the molecular space to medicinal properties, generative models are able to identify and optimize candidates for specific biological actions [1]. This new approach promises to save time, money, and resources, while yielding better results than traditional methods.

Since their introduction, a wide range of deep generative models have been proposed for molecular SMILES [5, 6], graphs [7–9], and point clouds [10], based on a variety of architectures, from variational autoencoders (VAEs) [11] and generative adversarial networks (GANs) [12] to normalizing flows [13] and denoising diffusion models [14]. In this work we focus on GANs and flow-based models, introducing MoFlowGAN a normalizing flow that can be trained adversarially and is heuristically biased towards generating high performing drug-like molecules as a tool for molecular discovery. Like its name, MoFlowGAN is an amalgamation of ideas from MoFlow [8], Flow-GAN [15], and MolGAN [7]; each being quite successful in their own respects we aim to draw from their strengths to make up for their weaknesses.

We adopt MolGAN's [7] reward network to heuristically bias generation towards high scoring molecules, while avoiding the mode collapse issues commonly seen with GANs by using a compound normalizing flow architecture in our generator allowing us to train MoFlowGAN on three objectives: likelihood maximization, adversarial loss minimization, and reward optimization. MoFlowGAN is, to our knowledge, the first hybrid FlowGAN model for graph-structured data.

[*]To whom correspondence should be addressed.

Working paper.

2 Preliminaries

We first recall key concepts and familiarize the reader with the notation used throughout the paper.

2.1 Normalizing Flows

A normalizing flow is an invertible transformation that $f_{\mathcal{Z}}(\mathbf{x}) : R^d \mapsto R^d$ that maps high-dimensional data $x \sim P_X(x)$ to a latent space representation $z \sim P_{\mathcal{Z}}(z)$, with inverse f^{-1} (i.e., $f^{-1} \circ f(z) = z$).[2] The distribution $P_{\mathcal{Z}}$ (called the "standard" or "base" distribution) is chosen to be one that is easy to sample from and whose density is easy to evaluate, typically an isotropic Gaussian.

$$\mathbf{z} = f_{\mathcal{Z}}(\mathbf{x}) = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{x}) \tag{1}$$

The resulting probability density $q(\mathbf{z})$, obtained by transformation of $\mathbf{x}$, having distribution q_0, by $f_{\mathcal{Z}}$, is evaluated through the *change of variable* formula.

$$\log q(\mathbf{z}) = \log q_0(\mathbf{x}) - \log |\det \nabla_{\mathbf{x}} f_{\mathcal{Z}}(\mathbf{x})| \tag{2}$$

However, computation of the Jacobian of high-dimensional distributions can be quite expensive. Thus we limit transformations f_n to those where the Jacobian is a triangular matrix, allowing us to evaluate the determinant with only the diagonal entries. We direct the reader to [16, 17] for the specifics of such transformations.

A flow can also be conditioned on some prior information $\mathbf{a} \in \mathbb{R}^D$. Such a *conditional normalizing flow* is denoted $f_{\mathcal{Z}|\mathbf{a}} : \mathbb{R}^d \times \mathbb{R}^D \mapsto \mathbb{R}^d$ [18]. The log-density in (2) accommodates this additional information giving

$$\log q(\mathbf{z}|\mathbf{a}) = \log q_0(\mathbf{x}|\mathbf{a}) - \log |\det \nabla_{\mathbf{x}} f_{\mathcal{Z}}(\mathbf{x}|\mathbf{a})| \tag{3}$$

A benefit of such transformations as $f_{\mathcal{Z}}$ is that expectations with respect to the transformed density can be calculated without explicit knowledge of $q(\mathbf{z})$ and without computation of the log-determinant Jacobian terms therefore making computation of the exact likelihood tractable [13, 19]. As a result a normalizing flow-based generative model is trained by maximizing the computed likelihood of the training samples (or equivalently, minimizing their negative log likelihood).

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}) \tag{4}$$

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [12] are implicit generative models that allow for inference without direct likelihood calculation consisting of two parts. The generator, a function $G_\theta : \mathbb{R}^k \mapsto \mathbb{R}^d$ takes a random latent vector $\mathbf{z} \in \mathbb{R}^k$ drawn from some tractable prior $p(\mathbf{z})$ and transforms it into a sample $G_\theta(\mathbf{z})$. The discriminator, a binary classifier $D_\phi : \mathbb{R}^d \mapsto \mathbb{R}$, takes both samples generated by the generator $G_\theta(\mathbf{z})$ and real training samples $\mathbf{x} \in \mathcal{D}$ and attempts to distinguish the true samples from the fake ones.

In the objective sense the discriminator (sometimes referred to as the critic) attempts to maximize the negative cross-entropy of its predictions while the generator attempts to minimize the same, leading G_θ learns from D_ϕ how to generate real and "life-like" samples. This classical minimax game is formalized

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x} \sim P_X} [\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim P_{\mathcal{Z}}} [\log (1 - D_\phi(G_\theta(\mathbf{z})))] \tag{5}$$

Since their inception much work has gone into improving the stability of GAN training [20]. In particular, Arjovsky et al. [21] propose constraining ϕ to the class of 1-Lipschitz functions ($\mathcal{F}$), reformulating the minimiax objective over the Wasserstein (also known as the earth-mover) distance between the model and data distributions.

$$\min_\theta \max_{\phi \in \mathcal{F}} \mathbb{E}_{\mathbf{x} \sim P_X} [D_\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P_{\mathcal{Z}}} [D_\phi(G_\theta(\mathbf{z}))] \tag{6}$$

Practically, this constraint is satisfied by clipping the discriminator weights, ϕ, or adding a gradient penalty term to the objective as proposed by Gulrajani et al. [22] (*vide infra*).

[2]We use $g \circ f(x)$ as shorthand for the composition $g(f(x))$

2.3 Deterministic Policy Gradients

In the traditional reinforcement learning problem an agent acts in a stochastic environment by choosing a series of actions over a set of time steps in order to maximize the total cumulative reward. This problem is modeled as a Markov Decision Process (MDP) consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, an initial state distribution with probability density $p_0(s_0)$, a conditional transition probability with density $p(s_{t+1}|s_t, a_t)$, and a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ [23, 24].

Policy gradient algorithms are broadly used in reinforcement learning problems with continuous action spaces [24]. A policy, $\pi_\psi : \mathcal{S} \mapsto \mathcal{P}(\mathcal{A})$ where $\mathcal{P}(\mathcal{A})$ is the set of probability measure on $\mathcal{A}$, guides the actions of the agent in the MDP whose goal is to maximize the cumulative total reward from start state s_0. Policies can be either stochastic ($\pi_\psi(s) = p\theta(a|s)$), where actions are probabilistically conditioned on environmental states, or deterministic ($\mu_\psi(s) = a$), in which actions are predetermined. In this work we focus on the latter, specifically, we utilize a simplified version of deep deterministic policy gradient introduced by Lillicrap et al. [25] where the reward is calculated by a neural network (R_ψ) that is trained to score molecular graphs with input from external chemoinformatics tools. We direct the reader to the work of De Cao and Kipf [7], specifically section 2.3, for the precise formulation of the reinforcement learning problem.

3 Related Work

Molecular graph generation. Deep generative models for molecular generation fall primarily into two regimes: those for string-based representations (e.g., SMILES [26], SELFIES [27], etc...) and those for molecular graphs. String-based models such as ORGAN [5], Grammar Variational Autoencoder (VAE) [28], Syntax Directed VAE [29], and SMILES-LSTM [30] operate on a formal grammar which describe molecules and molecular structure using an alphabet of characters that the model learns to reproduce, a goal similar to that of generative language models. Alternatively, models can operate directly in the space of molecular graphs such as MolGAN [7], Junction Tree VAE [31], NeVAE [32] and GraphAF [9]. We direct the reader to [33] for a detailed in-depth discussion of the various molecular representations and their respective strengths and weaknesses. In our work we represent molecules as undirected graphs where the nodes represent atoms and edges represent the bonds between them (further details are provided in section 4.1).

Molecular optimization. Several methods have been proposed for optimization of molecular generators for *de novo* drug design applications. Early approaches utilized generative learning [28–32, 34] in which a generator (usually a GAN or VAE) is coupled to a property predictor and the resulting optimization problem is solved using Bayesian optimization. Reinforcement learning-based optimization models [5, 7, 9, 35, 36] employ a similar workflow utilizing policy optimization algorithms to produce high-quality molecules. More recently, Jin et al. [37] introduced a graph-to-graph translation model based on matched molecular pair analysis [38] which learns to translate a molecular graph to a better molecular graph. Further, Zang and Wang [8] train a multi-layer perceptron to regress the latent representations of the molecules against the desired properties (penalized logP and QED), searching for the best-performing molecules through gradient ascent.

Normalizing flows. Normalizing flows have shown promising results on both image- [16, 17, 39] and graph-generation tasks [9, 40, 41]. GrapahAF [9] and MoFlow [8], to our knowledge, are the two flow-based models which achieve the current state-of-the-art performance. GraphAF is an autoregressive flow which relies on proximal policy optimization (PPO) [42] to build chemically valid molecular graphs in a stepwise (sequential) manner [9]. MoFlow is a normalizing flow that generates chemically valid molecules in a single shot using a conditional flow that learns to predict a molecule's atoms given a set of bonds. This single-shot inference/generation makes it more efficient than sequential models [8].

GANs. GANs have found notable applications [43] across a wide range of tasks including image/video processing and generation [44–46], computer vision [47–49], natural language processing [50, 51], and music generation [5, 52], but have been applied to the task of molecular generation to varying degrees of success. Guimaraes et al. [5] trained a seqGAN [52] model to generate text-based molecular representations (i.e., SMILES strings), incorporating an RL training mechanism based on REINFORCE [53] to target high-quality outputs. In the graph domain De Cao and Kipf [7] trained a

GAN to generate molecular graphs as bond and atom matrices using RelGCNs [54] and a simplified form of the deep deterministic policy gradients (DDPG) proposed by Lillicrap et al. [25] to bias generated molecules.

4 MoFlowGAN

MoFlowGAN consists of 3 components: a normalizing flow generator G_θ that learns to generate molecular graphs, a discriminator $\mathcal{D}_\phi$ that learns to distinguish real and fake molecular graphs, and a reward network $\mathcal{R}_\psi$ that learns to score molecular graphs. As alluded to earlier we represent molecules as bond tensor, b, and atom matrix, a, pairs, (a, b). We adopt MolGAN's [7] reward network to heuristically bias generation towards high performing molecules, while avoiding the mode collapse issues commonly seen with GANs by using a compound normalizing flow for the generator allowing us to train MoFlowGAN on three objectives: negative log likelihood, adversarial loss, and reward optimization.

4.1 Generator architecture

Based on the current state-of-the-art normalizing flow model MoFlow by Zang and Wang [8], our generator is itself a combination of two normalizing flow models, an invertible bond flow, $f_\mathcal{B}$, based on the Glow framework by Kingma and Dhariwal [39], and the graph conditional flow described by Zang and Wang [8], $f_{\mathcal{A}|\mathcal{B}}$. The bond flow learns bond tensors $b \in \mathcal{B} \subset \mathbb{R}^{c\times n\times n}$ while the graph conditional flow learns to generate the right atom matrix $a \in \mathcal{A} \subset \mathbb{R}^{n\times k}$ given b to create valid a molecule $m = (a, b) \in \mathcal{M} \subset \mathbb{R}^{n\times k+c\times n\times n}$. As characteristic of normalizing flow models the latent variables of both flows, $\mathcal{Z}_\mathcal{B} = f_\mathcal{B}$ and $\mathcal{Z}_{\mathcal{A}|\mathcal{B}}|\mathcal{B} = f_{\mathcal{A}|\mathcal{B}}$ follow an isotropic Gaussian distribution.

4.2 Discriminator and reward network architectures

Both discriminator $\mathcal{D}_\phi$ and reward $\mathcal{R}_\psi$ networks take a molecular graph (represented as atom matrices and bond tensors) as input and returns a scalar $\in (-\infty, \infty)$ for $\mathcal{D}$ and $\in [0, 1]$ for $\mathcal{R}$. We use the same architecture (based on Relational-GCN by Schlichtkrull et al. [54]) for both, but keep their parameters separate [7].

4.3 Learning objectives

We train MoFlowGAN on a mixture of likelihood, adversarial, and reward losses.

Generator. The generator loss is a weighted sum of negative log-likelihood (nll), Wasserstein GAN adversarial loss [21], and the reward network score for the fake molecule batch. The hyperparameters α and ρ represent the weights of the adversarial loss and reward score respectively; the weights of all objectives sum to one. The loss with respect to the generator is formalized

$$\mathcal{L}_G(m \in \mathcal{M}; \phi,\ \psi) = (1 - \alpha - \rho)\ \underbrace{\mathbb{E}_{m\sim P_\mathcal{M}}[-\log p_\theta(m)]}_{\text{nll}} - \alpha\ \underbrace{\mathcal{D}_\phi(G_\theta(z)}_{\text{WGAN loss}} - \rho\ \underbrace{\mathcal{R}_\psi(G_\theta(z))}_{\text{reward}} \tag{7}$$

Discriminator. To prevent undesired behaviors endemic of GANs (i.e., mode collapse and training instability) we adopt the Wasserstein-GAN (W-GAN) model [21] with mini-batch discrimination [20]. The discriminator is trained using the standard W-GAN loss with a slightly modified form of the gradient penalty described by Gulrajani et al. [22]. The discriminator loss is formalized

$$\mathcal{L}_\mathcal{D}(m \in \mathcal{M}, G_\theta(z); \phi) = \underbrace{-\mathcal{D}_\phi(m) + \mathcal{D}_\phi(G_\theta(z))}_{\text{WGAN loss}} + \underbrace{\lambda(\|\nabla_{\hat{x}}\mathcal{D}_\phi(\hat{x})\| - 1)^2}_{\text{gradient penalty}}, \tag{8}$$

where λ is a hyperparameter (we adopt λ = 10 from the original paper) and $\hat{x}$ is a mixed batch sampled from $m \sim \mathcal{M}$ and $G_\theta(z)$ where $z \sim \mathcal{Z}_B, \mathcal{Z}_{\mathcal{A}|\mathcal{B}}$.

Reward network. The reward network is trained to minimize the mean squared error between its predicted scores and those calculated using the open-source chemoinformatics toolkit RDKit [55] for both real and fake training batches.

$$\mathcal{L}_{\mathcal{R}}(m \in \mathcal{M}, G_\theta(z); \theta) = \text{MSE}[\mathcal{R}_\psi(G_\theta(z)),\ R(G_\theta(z))] + \text{MSE}[\mathcal{R}_\psi(m),\ R(m)] \tag{9}$$

Reward calculation. The reward network scores molecules on several distribution and chemical metrics. Each molecule is scored using the unweighted product of its individual metric scores which are normalized to [0,1]. Each molecule's reward score is built from

- Chemical validity, uniqueness, and novelty
- Diversity: fingerprint similarity of the molecule with those in the training set [7]
- Natural Product Likeness: structural similarity to complex natural products [56]
- LogP: compound lipophilicity
- QED: estimation of the molecule's drug-likeness [57]
- Synthetic Accessibility: ease of chemical synthesis[3] [58]

5 Experiments

We evaluate MoFlowGAN through three experiments:

- Objective trade-off
- Unconstrained generation
- Native property optimization

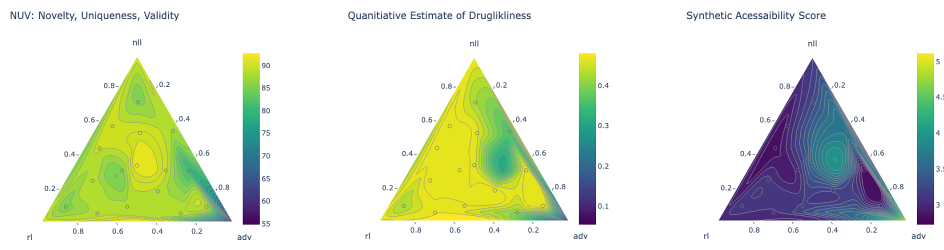
Baselines. We compare our model to several state-of-the-art molecular generation models. First, MoFlow by Zang and Wang [8] which is the native flow model that our generator is built upon and therefore equivalent to the objective described in (7) where $\alpha = \rho = 0$. We also compare our model performance to previous normalizing flow models Graph AF [9], GraphNVP [40], and Graph Residual Flow (GRF) [59]. In the realm of GANs and VAEs we compare our model to MolGAN [7], ORGAN [5] and NeVAE [32].

Dataset and preprocessing. For our experiments we use the QM9 small molecule dataset [60, 61]. QM9 contains 133k molecules with at most 9 heavy atoms (C, N, O, F) provided in SMILES representation. Following the example of Liu et al. [62], each molecule is kekulized using the popular chemoinformatics toolkit RDKit [55]. We then generate the atom matrices and bond tensors for each molecule by encoding each atom and bond using a one-hot scheme. Atom matrices (A) for molecules with fewer than 9 heavy atoms are padded with a "dummy" atom giving $A \in \mathbb{R}^{9\text{x}5}$. The bond tensors ($B$) are supplemented with an additional channel representing no bonds between atoms and dequantized [40] by adding uniform random noise ($\epsilon \sim U[0, 0.6]$) giving $B \in \mathbb{R}^{4\text{x}9\text{x}9}$.

Reward learning initialization. De Cao and Kipf [7] report divergence when training with any reward contribution to the generator loss during the early epochs of training. In our initial studies we observed quite similar results (accompanied by strikingly poor validity scores). To overcome these issues we follow the lead of De Cao and Kipf [7] and train the generator with no reward loss contribution for the first half of training.

Empirical Running Time. Using the setup described above, we implement MoFlowGAN using PyTorch-1.12. We train it using Adam [63] with a learning rate of 0.001, batch size of 256, β_1 of 0.5 using a single NVIDIA RTX A4000 GPU and 8 CPU cores through the Paperspace gradient platform. For the first 25 generator training iterations (approx. 4 epochs) we train the generator once for every 100 discriminator training iterations, afterwards we step back the training ratio to 8:1. Training on QM9 for 50 epochs takes approximately 3 hours, about 4x slower than the native MoFlow model [8]. We attribute this slowdown in training to calculation of the rewards which (to our knowledge) must be done sequentially on CPUs.

[3]Native SA scores are inverted for consistency with other *higher-is-better* metrics.



(a) NUV ↑ (b) QED ↑ (c) SA ↓

Figure 1: Ternary contour plots showing average NUV (a), QED (b), and SA (c) scores for molecules generated by MoFlowGAN models trained on variably weighted objectives.

Evaluation. We adopt widely used distribution learning metrics for generated datasets to evaluate our model, namely: **novelty**, the proportion of generated molecules which are not in the training set; **uniqueness**, the proportion of molecules that are not duplicated in a batch; **validity**, the proportion of generated graphs that represent chemically valid molecules [32]; and **diversity**, which measure the substructure similarity between generated molecules and the training data [7]. We additionally evaluate the compound metric **NUV** proposed by Zang and Wang [8] which measures the percentage of molecules that are novel, unique, valid. Finally, we evaluate two chemical metrics: **Quantitative Estimate of Drug-likeness (QED)** which scores molecules on much they structurally resemble real pharmaceuticals [57, 64]; and **synthetic accessibility** which measures how easily a molecule is to synthesized [58].

5.1 Objective trade-off

Setup. To determine the contributions of each individual objective, sixteen interpolation points (α, ρ) were generated randomly on the $K = 1$ ternary surface. Models were trained on QM9 for 20 epochs using these loss parameters and the setup described above. 10,000 samples were then generated (five batches of 2,000) and evaluated on the metrics described previously.

Results. Like other hybrid models [5, 7] we utilize the hyperparameters, α and ρ, to determine the relative contribution of each loss objective. Figure 2 shows the performance of 19 models trained with variably weighted objectives on both distribution (NUV) and chemical metrics (QED and synthetic accessibility). Full scores are reported in the appendices. In general, high performance models are clustered towards the center of the left edge in the low adversarial training regime. We select the best performing model, across both distribution and chemical metrics, with $\alpha = 0.1$ and $\rho = 0.42$ for subsequent experiments.

Of note is the lack of a clear trend towards higher scoring molecules with increasing reward learning contribution as reported in [7]. Instead the low scoring modes all lie in the low reward learning/high adversarial learning regime (the right axis of the ternary plots in Figure 2). We attribute this to two factors. First, GAN convergence occurs on a much longer time scale than learning by MLE and reward optimization i.e., the discriminator has not yet learned to distinguish between real and fake molecules well enough to for adversarial training to have substantial (more so than our other objectives) effect on generator performance. Second, this suggests that there is a similarity to likelihood learning and reward optimization. This is not altogether surprising since normalizing flows are trained to generate maximally probable outputs [65] which is an implicit form of our reward learning paradigm.

5.2 Unconstrained generation

Setup. To evaluate our model's ability to generate novel, unique, and valid molecular graphs relative to our baseline models we further train the model with the best performance across both distribution and chemical metrics ($\alpha = 0.27$, $\rho = 0.42$) for 50 epochs. 10,000 molecules were then generated and evaluated on the distribution metrics described above and compared to previously published results [8, 32] for the baseline models.

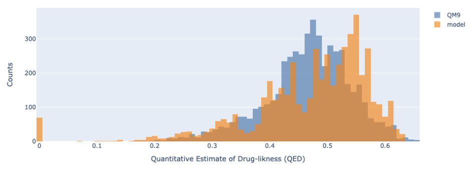
Table 1: Unconstrained molecular generation on QM9

Class	**Model**	**% Novel ↑**	**% Unique ↑**	**% Valid ↑**	**% NUV ↑**
GANs	MolGAN	94.20	10.40	**98.10**	9.61
	ORGAN	–	97.20	96.10	–
VAEs	NeVAE†	**100.00**	67.60	68.20	46.10
Flows	GraphNVP	58.20	99.20	83.10	47.98
	GRF	58.60	66.00	84.50	32.68
	GraphAF	88.83	94.51	67.00	56.25
	MoFlow†‡	62.92	99.38	90.90	56.84
	MoFlowGAN†	68.54	**100.00**	97.08	**66.54**

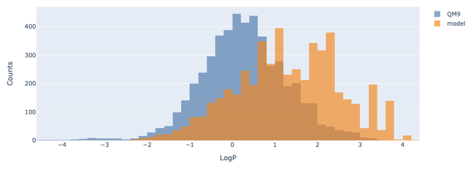
†Results without *post hoc* validity correction (NeVAE calls this "masking")
‡Results obtained for MoFlow after only 50 training epochs

Results. We report our results in Table 1. We observe that training our model for an additional 30 epochs improves validity scores significantly (over 10 percentage points), but not novelty scores (which show an improvement of only 2 percentage points). We note that flow models are particularly adept at generating unique molecules as they consistently score higher than unmasked VAEs and GANs (n.b. ORGAN [5] optimizes for unique molecules through its own reward network.)

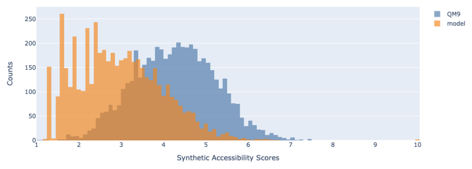
Here we must unbury the lead. MoFlowGAN achieves the highest compound distribution performance among all baseline models with 66.5% of generated molecules being novel, unique, and chemically valid. In fact, after 50 epochs of training MoFlowGAN outperforms native parent model, MoFlow. Admittedly, MoFlow achieves near perfect performance (93.5% NUV) after 200 epochs of training on QM9 [8]. We note that the major shortfall of our model is the low novelty score; we posit that weighting these distribution metrics more heavily in the early stages of training or using a RL component with stronger gradient signal [53] will allow us to be competitive with the current state-of-the-art generative models on this front.



(a) QED ↑



(b) logP ↑



(c) SA ↓

Figure 2: Distribution of QED (a), logP (b), and synthetic accessibility (c) scores for 10,000 molecules drawn from QM9 (blue) and generated using fully trained MoFlowGAN (orange).

5.3 Native property optimization

Setup. We trained the model with the best performance across both distribution and chemical metrics ($\alpha = 0.27$, $\rho = 0.42$) for 50 epochs. Using the fully trained model, we randomly generate and score 10,000 molecules. For comparison we randomly draw 10,000 molecules from QM9 without replacement and score them using the same method as the generated samples.

Results. Finally we evaluate the ability of the reward training component to guide molecular generation by comparing the chemical trait distribution of 10,000 generated molecules to that of 10,000 real samples from QM9. An "undirected" model learns to produce molecules that match the distribution of the dataset [66, 67] our results (Figure 2) demonstrate that MoFlowGAN is biased toward generating higher quality molecules than those it was trained on. We attribute this to the reward learning objective. We note that this property optimization is innate to MoFlowGAN and does not require an additional supervised learning step as in the original MoFlow model [8].

6 Conclusion

In this paper, we propose a tripartite deep graph generative model, MoFlowGAN, for targeted molecular graph generation. MoFlowGAN consists of a normalizing flow generator that learns invertible mappings for bond networks and atoms, a GCN-based discriminator which allows for adversarial training, and a modular reward network that can be used to bias the generator towards "improved outputs". MoFlowGAN is competitive with recent state-of-the-art models including its parent architectures. Future work will explore MoFlowGAN's performance on larger datasets with more druglike molecules, such as ZINC250 [68] as well as tradeoff of between learning objectives and model depth.

Data availability

The code for this paper is available at `https://github.com/thisisntnathan/MoFlowGAN`.

Acknowledgments and Disclosure of Funding

The authors would like to thank Prof. Kilian Weinberger (Cornell) for insights, feedback, and fruitful discussion. N.M.L. thanks Prof. David B. Collum (Cornell) for time and financial support.

Author contributions

N.M.L. conceived of the project, designed the experiments, contributed code, and wrote the manuscript. M.D.L. and M.T.F. contributed some code and gave approval to the final manuscript.

References

[1] Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F. Jensen. Generative models for molecular discovery: Recent advances and challenges. *WIREs Computational Molecular Science*, 12(5):e1608, 2022.

[2] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[3] Quentin Vanhaelen, Yen-Chu Lin, and Alex Zhavoronkov. The Advent of Generative Chemistry. *ACS Medicinal Chemistry Letters*, 11(8):1496–1505, 2020.

[4] Abdulelah S. Alshehri, Rafiqul Gani, and Fengqi You. Deep learning and knowledge-based methods for computer-aided molecular design—toward a unified approach: State-of-the-art and future directions. *Computers & Chemical Engineering*, 141:107005, 2020.

[5] Gabriel L. Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv preprint arXiv:1705.10843*, 2017.

[6] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L. Guimaraes, and Alan Aspuru-Guzik. Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). *ChemRxiv preprint chemrxiv.5309668.v3*, 2017.

[7] Nicola De Cao and Thomas N. Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[8] Chengxi Zang and Fei Wang. MoFlow: An Invertible Flow Model for Generating Molecular Graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 617–626. Association for Computing Machinery, 2020.

[9] Chence Shi*, Minkai Xu*, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. *International Conference on Learning Representations*, 2020.

[10] Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant Diffusion for Molecule Generation in 3D. *International Conference on Machine Learning*, pages 8867–8887, 2022.

[11] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, 27, 2014.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv preprint arXiv:1406.2661*, 2014.

[13] Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *International Conference on Machine Learning*, pages 1530–1538, 2015.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[15] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[16] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017.

[18] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning Likelihoods with Conditional Normalizing Flows. *arXiv preprint arXiv:1912.00042*, 2019.

[19] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217 – 233, 2010.

[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems*, 29, 2016.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. *International Conference on Machine Learning*, pages 214–223, 2017.

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems*, 30, 2017.

[23] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[24] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. *International Conference on Machine Learning*, pages 387–395, 2014.

[25] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[26] 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.

[27] Mario Krenn, Florian Häse, Akshat Kumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, oct 2020.

[28] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar Variational Autoencoder. *International Conference on Machine Learning*, pages 1945–1954, 2017.

[29] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-Directed Variational Autoencoder for Structured Data. *International Conference on Learning Representations*, 2018.

[30] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Central Science*, 4(1):120–131, 2018.

[31] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning*, pages 2323–2332, 2018.

[32] Bidisha Samanta, Abir De, Gourhari Jana, Vicenç Gómez, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. NeVAE: A Deep Generative Model for Molecular Graphs. *Journal of Machine Learning Research*, 21(114):1–33, 2020.

[33] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in AI-driven drug discovery: A review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.

[34] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, 2018.

[35] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *Advances in Neural Information Processing Systems*, 31, 2018.

[36] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, 2018.

[37] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning Multimodal Graph-to-Graph Translation for Molecule Optimization. *International Conference on Learning Representations*, 2019.

[38] Ed Griffen, Andrew G. Leach, Graeme R. Robb, and Daniel J. Warner. Matched Molecular Pairs as a Medicinal Chemistry Tool. *Journal of Medicinal Chemistry*, 54(22):7739–7750, 2011.

[39] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *Advances in Neural Information Processing Systems*, 31, 2018.

[40] Madhawa Kaushalya, Ishiguro Katushiko, Nakago Kosuke, and Abe Motoki. GraphNVP: An Invertible Flow Model for Generating Molecular Graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[41] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph Normalizing Flows. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[44] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[45] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv preprint arXiv:1710.10196*, 2017.

[46] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[47] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual Generative Adversarial Networks for Small Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

[48] Kiana Ehsani, Roozbeh Mottaghi, and Ali Farhadi. SeGAN: Segmenting and Generating the Invisible. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[49] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221, 2018.

[50] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial Ranking for Language Generation. *Advances in Neural Information Processing Systems*, 30, 2017.

[51] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. MirrorGAN: Learning Text-to-image Generation by Redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1505–1514, 2019.

[52] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[53] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[54] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. *European Semantic Web Conference*, pages 593–607, 2018.

[55] RDKit, online. RDKit: Open-source cheminformatics. `http://www.rdkit.org`, 2022. [Online; accessed 01-Dec-2022].

[56] Peter Ertl, Silvio Roggo, and Ansgar Schuffenhauer. Natural Product-likeness Score and Its Application for Prioritization of Compound Libraries. *Journal of Chemical Information and Modeling*, 48(1):68–74, 2008.

[57] G. Richard Bickerton, Gaia V. Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98, 2012.

[58] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:1–11, 2009.

[59] Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. Graph Residual Flow for Molecular Graph Generation. *arXiv preprint arXiv:1909.13521*, 2019.

[60] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[61] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[62] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained Graph Variational Autoencoders for Molecule Design. *Advances in Neural Information Processing Systems*, 31, 2018.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] Scott A. Wildman and Gordon M. Crippen. Prediction of Physicochemical Parameters by Atomic Contributions. *Journal of Chemical Information and Computer Sciences*, 39(5):868–873, 1999.

[65] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11): 3964–3979, 2021.

[66] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[67] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, 11, 2020.

[68] Teague Sterling and John J. Irwin. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Additional experimental data

A Objective trade-off scores

Table 2: Generation on QM9 - Distribution metrics

Loss contribution			Metric				
nll	**adv**	**rl**	**% Novel ↑**	**% Unique ↑**	**% Valid ↑**	**% NUV ↑**	**% Diversity ↑**
0.34	0.33	0.33	69.16	99.96	94.64	65.43	77.13
0.1	0.09	0.81	45.20	96.75	92.62	40.50	86.14
0.25	0.15	0.6	55.26	98.85	91.28	49.86	81.40
0.44	0.09	0.47	46.22	97.02	92.60	41.52	86.77
0.57	0.09	0.34	56.28	98.53	91.68	50.84	84.41
0.71	0.15	0.14	55.70	98.78	89.84	49.43	83.90
0.31	**0.27**	**0.42**	64.88	98.81	87.28	55.95	83.57
0.53	0.25	0.22	70.96	99.96	89.62	63.57	76.41
0.1	0.4	0.5	78.80	99.55	90.58	71.06	80.28
0.31	0.49	0.2	64.76	99.46	90.16	58.07	80.47
0.54	0.41	0.05	59.50	99.36	91.02	53.81	81.79
0.06	0.69	0.25	53.32	99.32	91.02	48.20	80.82
0.31	0.61	0.08	61.14	97.63	83.74	49.99	86.84
0.1	0.8	0.1	58.92	98.57	91.36	53.06	83.97
0.06	0.27	0.67	63.60	98.99	90.32	56.86	86.07
0.19	0.51	0.3	63.66	99.52	86.68	54.91	78.20

Table 3: Generation on QM9 - Chemical metrics

Loss contribution			Metric				
nll	**adv**	**rl**	**NP Score ↑**	**logP ↑**	**SA Score ↓**	**QED ↑**	**DC Score ↑**
0.34	0.33	0.33	1.068	0.564	3.498	0.457	1.16
0.1	0.09	0.81	0.954	1.657	2.693	0.482	0.88
0.25	0.15	0.6	1.096	1.396	3.039	0.473	1.00
0.44	0.09	0.47	1.042	1.896	2.759	0.481	0.89
0.57	0.09	0.34	0.976	1.283	2.903	0.472	0.96
0.71	0.15	0.14	1.010	1.394	2.964	0.472	0.98
0.31	**0.27**	**0.42**	0.945	1.320	2.969	0.470	1.00
0.53	0.25	0.22	1.066	0.622	3.707	0.382	1.21
0.1	0.4	0.5	1.126	0.988	3.919	0.290	1.28
0.31	0.49	0.2	1.125	1.144	3.535	0.388	1.15
0.54	0.41	0.05	1.055	1.166	3.108	0.471	1.03
0.06	0.69	0.25	1.126	1.182	3.085	0.474	1.02
0.31	0.61	0.08	0.924	1.522	2.752	0.467	0.93
0.1	0.8	0.1	1.059	1.354	2.991	0.475	0.99
0.06	0.27	0.67	1.019	1.108	3.020	0.463	1.01
0.19	0.51	0.3	1.186	1.169	3.456	0.412	1.13

B Unconstrained generation on QM9

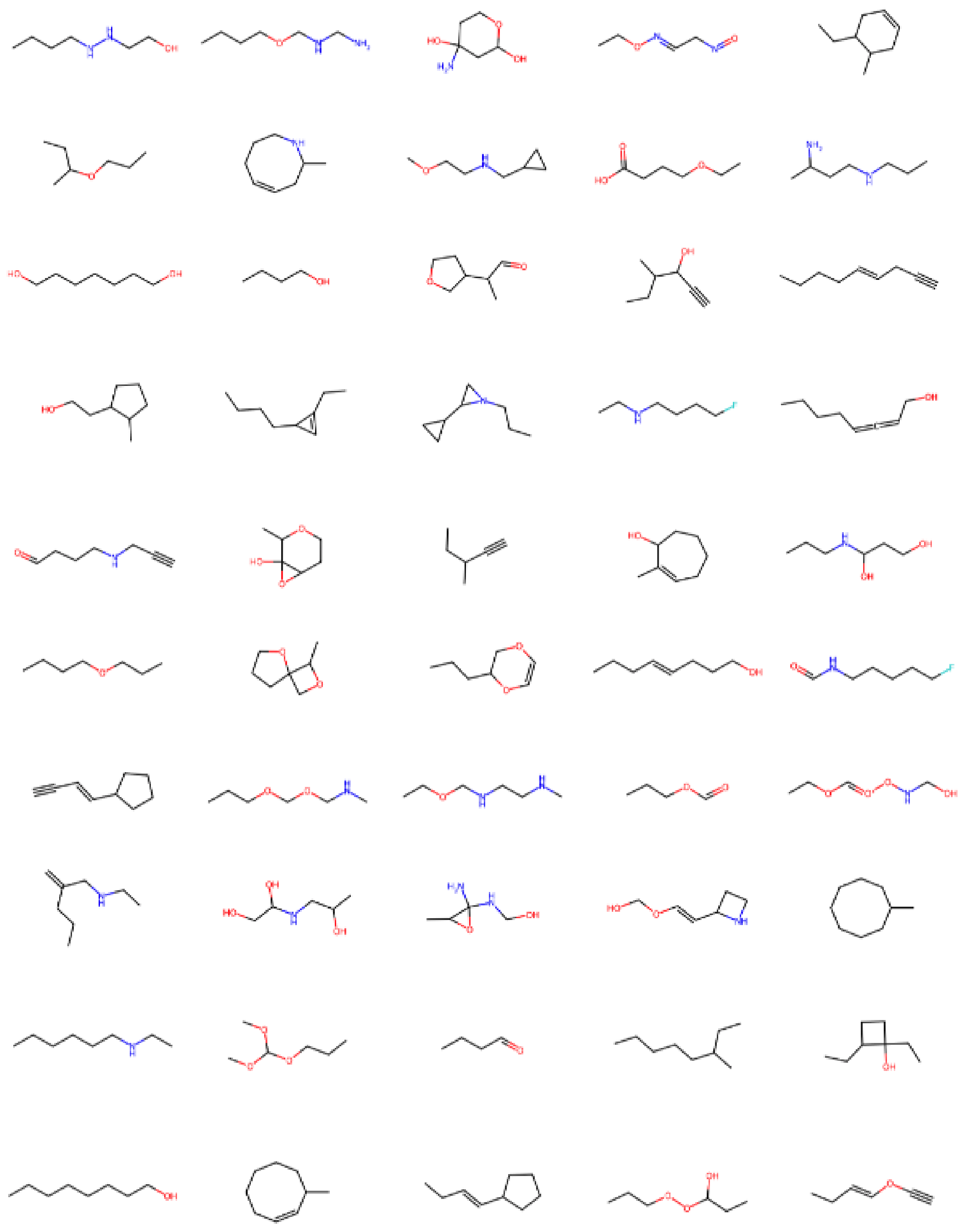


Figure 3: Samples generated with MoFlowGAN trained on QM9