MolGAN: An implicit generative model for small molecular graphs

Nicola De Cao [1] **Thomas Kipf** [1]

Abstract

Deep generative models for graph-structured data offer a new angle on the problem of chemical synthesis: by optimizing differentiable models that directly generate molecular graphs, it is possible to side-step expensive search procedures in the discrete and vast space of chemical structures. We introduce MolGAN, an implicit, likelihood-free generative model for small molecular graphs that circumvents the need for expensive graph matching procedures or node ordering heuristics of previous likelihood-based methods. Our method adapts generative adversarial networks (GANs) to operate directly on graph-structured data. We combine our approach with a reinforcement learning objective to encourage the generation of molecules with specific desired chemical properties. In experiments on the QM9 chemical database, we demonstrate that our model is capable of generating close to 100% valid compounds. MolGAN compares favorably both to recent proposals that use string-based (SMILES) representations of molecules and to a likelihood-based method that directly generates graphs, albeit being susceptible to mode collapse.

1. Introduction

Finding new chemical compounds with desired properties is a challenging task with important applications such as *de novo* drug design (Schneider & Fechner, 2005). The space of synthesizable molecules is vast and search in this space proves to be very difficult, mostly owing to its discrete nature.

Recent progress in the development of deep generative models has spawned a range of promising proposals to address this issue. Most works in this area (Gómez-Bombarelli et al., 2016; Kusner et al., 2017; Guimaraes et al., 2017; Dai et al., 2018) make use of a so-called SMILES representation (Weininger, 1988) of molecules: a string-based representation derived from molecular graphs. Recurrent neural networks (RNNs) are ideal candidates for these representations and consequently, most recent works follow the recipe of applying RNN-based generative models on this type of encoding. String-based representations of molecules, however, have certain disadvantages: RNNs have to spend capacity on learning both the syntactic rules and the order ambiguity of the representation. Besides, this is approach not applicable to generic (non-molecular) graphs.

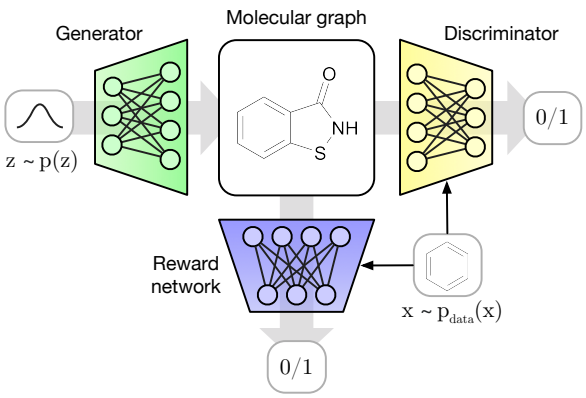


Figure 1. Schema of MolGAN. A vector $\boldsymbol{z}$ is sampled from a prior and passed to the generator which outputs the graph representation of a molecule. The discriminator classifies whether the molecular graph comes from the generator or the dataset. The reward network tries to estimate the reward for the chemical properties of a particular molecule provided by an external software.

SMILES strings are generated from a graph-based representation of molecules, thereby working in the original graph space has the benefit of removing additional overhead. With recent progress in the area of deep learning on graphs (Bronstein et al., 2017; Hamilton et al., 2017), training deep generative models directly on graph representations becomes a feasible alternative that has been explored in a range of recent works (Kipf & Welling, 2016b; Johnson, 2017; Grover et al., 2017; Li et al., 2018b; Simonovsky & Komodakis, 2018; You et al., 2018).

Likelihood-based methods for molecular graph generation (Li et al., 2018b; Simonovsky & Komodakis, 2018) however, either require providing a fixed (or randomly chosen) ordered representation of the graph or an expensive graph

[1]Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands. Correspondence to: Nicola De Cao <nicola.decao@gmail.com>.

matching procedure to evaluate the likelihood of a generated molecule, as the evaluation of all possible node orderings is prohibitive already for graphs of small size.

In this work, we sidestep this issue by utilizing implicit, likelihood-free methods, in particular, a generative adversarial network (GAN) (Goodfellow et al., 2014) that we adapt to work directly on graph representations. We further utilize a reinforcement learning (RL) objective similar to ORGAN (Guimaraes et al., 2017) to encourage the generation of molecules with particular properties.

Our molecular GAN (MolGAN) model (outlined in Figure 1) is the first to address the generation of graph-structured data in the context of molecular synthesis using GANs (Goodfellow et al., 2014). The generative model of MolGAN predicts discrete graph structure at once (i.e., non-sequentially) for computational efficiency, although sequential variants are possible in general. MolGAN further utilizes a permutation-invariant discriminator and reward network (for RL-based optimization towards desired chemical properties) based on graph convolution layers (Bruna et al., 2013; Duvenaud et al., 2015; Kipf & Welling, 2016a; Schlichtkrull et al., 2017) that both operate directly on graph-structured representations.

2. Background

2.1. Molecules as graphs

Most previous deep generative models for molecular data (Gómez-Bombarelli et al., 2016; Kusner et al., 2017; Guimaraes et al., 2017; Dai et al., 2018) resort to generating SMILES representations of molecules. The SMILES syntax, however, is not robust to small changes or mistakes, which can result in the generation of invalid or drastically different structures. Grammar VAEs (Kusner et al., 2017) alleviate this problem by constraining the generative process to follow a particular grammar.

Operating directly in the space of graphs has recently been shown to be a viable alternative for generative modeling of molecular data (Li et al., 2018b; Simonovsky & Komodakis, 2018) with the added benefit that all generated outputs are valid graphs (but not necessarily valid molecules).

We consider that each molecule can be represented by an undirected graph $\mathcal{G}$ with a set of edges $\mathcal{E}$ and nodes $\mathcal{V}$. Each atom corresponds to a node $v_i \in \mathcal{V}$ that is associated with a T-dimensional one-hot vector $\mathbf{x}_i$, indicating the type of the atom. We further represent each atomic bond as an edge $(v_i, v_j) \in \mathcal{E}$ associated with a bond type $y \in \{1, ..., Y\}$. For a molecular graph with N nodes, we can summarize this representation in a node feature matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T \in \mathbb{R}^{N \times T}$ and an adjacency tensor $\mathbf{A} \in \mathbb{R}^{N \times N \times Y}$ where $\mathbf{A}_{ij} \in \mathbb{R}^Y$ is a one-hot vector indicating the type of the edge between i and j.

2.2. Implicit vs. likelihood-based methods

Likelihood-based methods such as the variational auto-encoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) typically allow for easier and more stable optimization than implicit generative models such as a GAN (Goodfellow et al., 2014). When generating graph-structured data, however, we wish to be invariant to reordering of nodes in the (ordered) matrix representation of the graph, which requires us to either perform a prohibitively expensive graph matching procedure (Simonovsky & Komodakis, 2018) or to evaluate the likelihood for all possible node permutations explicitly.

By resorting to implicit generative models, in particular to the GAN framework, we circumvent the need for an explicit likelihood. While the discriminator of the GAN can be made invariant to node ordering by utilizing graph convolutions (Bruna et al., 2013; Duvenaud et al., 2015; Kipf & Welling, 2016a) and a node aggregation operator (Li et al., 2016), the generator still has to decide on a specific node ordering when generating a graph. Since we do not provide a likelihood, however, the generator is free to choose any suitable ordering for the task at hand. We provide a brief introduction to GANs in the following.

Generative adversarial networks GANs (Goodfellow et al., 2014) are implicit generative models in the sense that they allow for inference of model parameters without requiring one to specify a likelihood.

A GAN consist of two main components: a generative model G_θ, that learns a map from a prior to the data distribution to sample new data-points, and a discriminative model D_ϕ, that learns to classify whether samples came from the data distribution rather than from G_θ. Those two models are implemented as neural networks and trained simultaneously with stochastic gradient descent (SGD). G_θ and D_ϕ have different objectives, and they can be seen as two players in a minimax game

$$\min_\theta \max_\phi \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D_\phi(G_\theta(\boldsymbol{z}))] \,, \quad (1)$$

where G_θ tries to generate samples to fool the discriminator and D_ϕ tries to differentiate samples correctly. To prevent undesired behaviour such as mode collapse (Salimans et al., 2016) and to stabilize learning, we use minibatch discrimination (Salimans et al., 2016) and improved WGAN (Gulrajani et al., 2017), an alternative and more stable GAN model that minimizes a better suited divergence.

Improved WGAN WGANs (Arjovsky et al., 2017) minimize an approximation of the Earth Mover (EM) distance

(also know as Wasserstein-1 distance) defined between two probability distributions. Formally, the Wasserstein distance between p and q, using the Kantorovich-Rubinstein duality is

$$D_W[p||q] = \frac{1}{K} \sup_{\|f\|_L < K} \mathbb{E}_{x\sim p(x)}\big[f(x)\big] - \mathbb{E}_{x\sim q(x)}\big[f(x)\big] \ , \tag{2}$$

where in the case of WGAN, p is the empirical distribution and q is the generator distribution. Note that the supremum is over all the K-Lipschitz functions for some $K > 0$.

Gulrajani et al. (2017) introduce a gradient penalty as an alternative soft constraint on the 1-Lipschitz continuity as an improvement upon the *gradient clipping* scheme from the original WGAN. The loss with respect to the generator remains the same as in WGAN, but the loss function with respect to the discriminator is modified to be

$$L(\boldsymbol{x}^{(i)}, G_\theta(\boldsymbol{z}^{(i)}); \phi) = \underbrace{-D_\phi(\boldsymbol{x}^{(i)}) + D_\phi(G_\theta(\boldsymbol{z}^{(i)}))}_{\text{original WGAN loss}} + \underbrace{\alpha \left(\|\nabla_{\hat{\boldsymbol{x}}^{(i)}} D_\phi(\hat{\boldsymbol{x}}^{(i)})\| - 1 \right)^2}_{\text{gradient penalty}} , \tag{3}$$

where α is a hyperparameter (we use $\alpha = 10$ as in the original paper), $\hat{\boldsymbol{x}}^{(i)}$ is a sampled linear combination between $\boldsymbol{x}^{(i)} \sim p_{data}(\boldsymbol{x})$ and $G_\theta(\boldsymbol{z}^{(i)})$ with $\boldsymbol{z}^{(i)} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$, thus $\hat{\boldsymbol{x}}^{(i)} = \epsilon\, \boldsymbol{x}^{(i)} + (1-\epsilon)\, G_\theta(\boldsymbol{z}^{(i)})$ with $\epsilon \sim \mathcal{U}(0,1)$.

2.3. Deterministic policy gradients

A GAN generator learns a transformation from a prior distribution to the data distribution. Thus, generated samples resemble data samples. However, in *de novo* drug design methods, we are not only interested in generating chemically valid compounds, but we want them to have some useful property (e.g., to be easily synthesizable). Therefore, we also optimize the generation process towards some non-differentiable metrics using reinforcement learning.

In reinforcement learning, a stochastic policy is represented by $\pi_\theta(s) = p_\theta(a|s)$ which is a parametric probability distribution in θ that selects a categorical action a conditioned on an environmental state s. Conversely, a deterministic policy is represented by $\mu_\theta(s) = a$ which deterministically outputs an action.

In initial experiments, we explored using REINFORCE (Williams, 1992) in combination with a stochastic policy that models graph generation as a set of categorical choices (actions). However, we found that it converged poorly due to the high dimensional action space when generating graphs at once. We instead base our method on a deterministic policy gradient algorithm which is known to perform well in high-dimensional action spaces (Silver et al., 2014). In particular, we employ a version of deep deterministic policy gradient (DDPG) introduced by Lillicrap et al. (2015), an off-policy actor-critic algorithm that uses deterministic policy gradients to maximize an approximation of the expected future reward.

In our case, the policy is the GAN generator G_θ which takes a sample $\boldsymbol{z}$ for the prior as input, instead of an environmental state s, and it outputs a molecular graph as an action ($a = \mathcal{G}$). Moreover, we do not model episodes, so there is no need to assess the quality of a state-action combination since it does only depend on the graph $\mathcal{G}$. Therefore, we introduce a learnable and differentiable approximation of the reward function $\hat{R}_\psi(\mathcal{G})$ that predicts the immediate reward, and we train it via a mean squared error objective based on the real reward provided by an external system (e.g., the synthesizability score of a molecule). Then, we train the generator maximizing the predicted reward via $\hat{R}_\psi(\mathcal{G})$ which, being differentiable, provides a gradient to the policy towards the desired metric. Notice that, differently form DDPG, we do not use experience replay or target networks (see original work).

3. Model

The MolGAN architecture (Figure 2) consists of three main components: a generator G_θ, a discriminator D_ϕ and a reward network $\hat{R}_\psi$.

The generator takes a sample from a prior distribution and generates an annotated graph $\mathcal{G}$ representing a molecule. Nodes and edges of $\mathcal{G}$ are associated with annotations denoting atom type and bond type respectively. The discriminator takes both samples from the dataset and the generator and learns to distinguish them. Both G_θ and D_ϕ are trained using improved WGAN such that the generator learns to match the empirical distribution and eventually outputs valid molecules.

The reward network is used to approximate the reward function of a sample and optimize molecule generation towards non-differentiable metrics using reinforcement learning. Dataset and generated samples are inputs of $\hat{R}_\psi$, but, differently from the discriminator, it assigns scores to them (e.g., how likely the generated molecule is to be soluble in water). The reward network learns to assign a reward to each molecule to match a score provided by an external software[1]. Notice that, when MolGAN outputs a non-valid molecule, it is not possible to assign a reward since the graph is not even a compound. Thus, for invalid molecular graphs, we assign zero rewards.

The discriminator is then trained using the WGAN objective while the generator uses a linear combination of the WGAN

[1]We used the RDKit Open-Source Cheminformatics Software: http://www.rdkit.org.

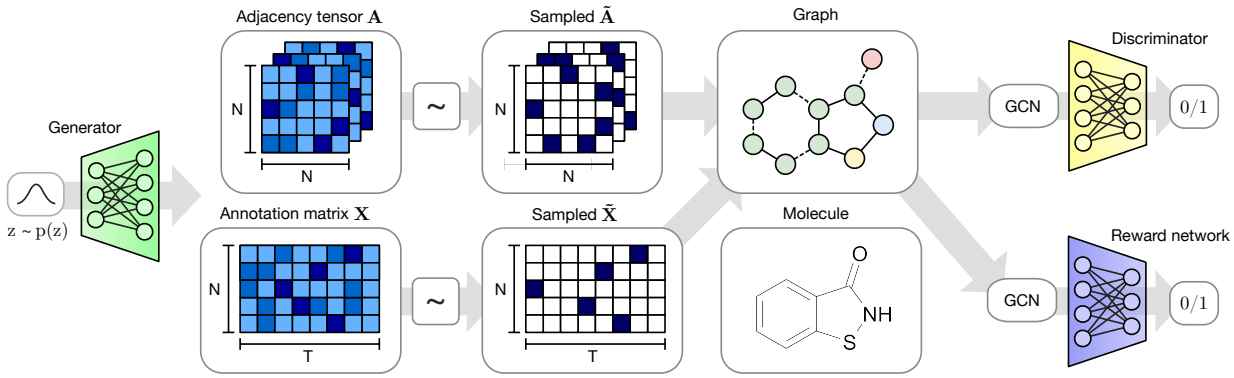


Figure 2. Outline of MolGAN. From *left*: the generator takes a sample from a prior distribution and generates a dense adjacency tensor $\boldsymbol{A}$ and an annotation matrix $\boldsymbol{X}$. Subsequently, sparse and discrete $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ are obtained from $\boldsymbol{A}$ and $\boldsymbol{X}$ respectively via categorical sampling. The combination of $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ represents an annotated molecular graph which corresponds to a specific chemical compound. Finally, the graph is processed by both the discriminator and reward networks that are invariant to node order permutations and based on Relational-GCN (Schlichtkrull et al., 2017) layers.

loss and the RL loss:

$$L(\theta) = \lambda \cdot L_{WGAN} + (1 - \lambda) \cdot L_{RL} \ , \tag{4}$$

where $\lambda \in [0, 1]$ is a hyperparameter that regulates the trade-off between the two components.

3.1. Generator

$G_\phi(\boldsymbol{z})$ takes D-dimensional vectors $\boldsymbol{z} \in \mathbb{R}^D$ sampled from a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and outputs graphs. While recent works have shown that it is feasible to generate graphs of small size by using an RNN-based generative model (Johnson, 2017; You et al., 2018; Li et al., 2018a;b) we, for simplicity, utilize a generative model that predicts the entire graph at once using a simple multi-layer perceptron (MLP), as similarly done in Simonovsky & Komodakis (2018). While this limits our study to graphs of a pre-chosen maximum size, we find that it is significantly faster and easier to optimize.

We restrict the domain to graphs of a limited number of nodes and, for each $\boldsymbol{z}$, G_θ outputs two continuous and dense objects: $\boldsymbol{X} \in \mathbb{R}^{N \times T}$ that defines atom types and $\boldsymbol{A}^{N \times N \times Y}$ that defines bonds types (see Section 2.1). Both $\boldsymbol{X}$ and $\boldsymbol{A}$ have a probabilistic interpretation since each node and edge type is represented with probabilities of categorical distributions over types. To generate a molecule we obtain discrete, sparse objects $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{A}}$ via categorical sampling from $\boldsymbol{X}$ and $\boldsymbol{A}$, respectively. We overload notation and also represent samples from the dataset with binary $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{A}}$.

As this discretization process is non-differentiable, we explore three model variations to allow for gradient-based training: We can i) use the continuous objects $\boldsymbol{X}$ and $\boldsymbol{A}$ directly during the forward pass (i.e., $\tilde{\boldsymbol{X}} = \boldsymbol{X}$ and $\tilde{\boldsymbol{A}} = \boldsymbol{A}$), ii) add Gumbel noise to $\boldsymbol{X}$ and $\boldsymbol{A}$ before passing them to D_ϕ and $\hat{R}_\psi$ in order to make the generation stochastic while still forwarding continuous objects (i.e., $\tilde{\boldsymbol{X}}_{ij} = \boldsymbol{X}_{ij} + \text{Gumbel}(\mu = 0, \beta = 1)$ and $\tilde{\boldsymbol{A}} = \boldsymbol{A}_{ijy} + \text{Gumbel}(\mu = 0, \beta = 1)$), or iii) use a straight-through gradient based on categorical reparameterization with the Gumbel-Softmax (Jang et al., 2016; Maddison et al., 2016), taht is we use a sample form a categorical distribution during the forward pass (i.e., $\tilde{\boldsymbol{X}}_i = \text{Cat}(\boldsymbol{X}_i)$ and $\tilde{\boldsymbol{A}}_{\boldsymbol{ij}} = \text{Cat}(\boldsymbol{A}_{ij})$) and the continuous relaxed values (i.e., the original $\boldsymbol{X}$ and $\boldsymbol{A}$) in the backward pass.

3.2. Discriminator and reward network

Both the discriminator D_ϕ and the reward network $\hat{R}_\psi$ receive a graph as input, and they output a scalar value each. We choose the same architecture for both networks but do not share parameters between them. A series of graph convolution layers convolve node signals $\tilde{\boldsymbol{X}}$ using the graph adjacency tensor $\tilde{\boldsymbol{A}}$. We base our model on Relational-GCN (Schlichtkrull et al., 2017), a convolutional network for graphs with support for multiple edge types. At every layer, feature representations of nodes are convolved/propagated according to:

$$\boldsymbol{h}_i^{\prime(\ell+1)} = f_s^{(\ell)}(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{x}_i) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{\tilde{\boldsymbol{A}}_{ijy}}{|\mathcal{N}_i|} f_y^{(\ell)}(\boldsymbol{h}_j^{(\ell)}, \boldsymbol{x}_i) \ ,$$
$$\boldsymbol{h}_i^{(\ell+1)} = \tanh(\boldsymbol{h}_i^{\prime(\ell+1)}) \ , \tag{5}$$

where $\boldsymbol{h}_i^{(\ell)}$ is the signal of the node i at layer ℓ and $f_s^{(\ell)}$ is a linear transformation function that acts as a self-connection between layers. We further utilize an edge type-specific affine function $f_y^{(\ell)}$ for each layer. $\mathcal{N}_i$ denotes the set of neighbors for node i. The normalization factor $1/|\mathcal{N}_i|$ ensures that activations are on a similar scale irrespective of the number of neighbors.

After several layers of propagation via graph convolutions, following Li et al. (2016) we aggregate node embeddings into a graph level representation vector as

$$\boldsymbol{h}'_{\mathcal{G}} = \sum_{v \in \mathcal{V}} \sigma(i(\boldsymbol{h}_v^{(L)}, \boldsymbol{x}_v)) \odot \tanh(j(\boldsymbol{h}_v^{(L)}, \boldsymbol{x}_v)) \ ,$$
$$\boldsymbol{h}_{\mathcal{G}} = \tanh \boldsymbol{h}'_{\mathcal{G}} \ , \tag{6}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the logistic sigmoid function, i and j are MLPs with a linear output layer and $\odot$ denotes element-wise multiplication. Then, $\boldsymbol{h}_{\mathcal{G}}$ is a vector representation of the graph $\mathcal{G}$ and it is further processed by an MLP to produce a graph level scalar output $\in (-\infty, +\infty)$ for the discriminator and $\in (0, 1)$ for the reward network.

4. Related work

Objective-Reinforced Generative Adversarial Networks (ORGAN) by Guimaraes et al. (2017) is the closest related work to ours. Their model relies on SeqGAN (Yu et al., 2017) to adversarially learn to output sequences while optimizing towards chemical metrics with REINFORCE (Williams, 1992). The main differences from our approach is that they model sequences of SMILES as molecular representations instead of graphs, and their RL component uses REINFORCE while we use DDPG. Segler et al. (2017) also employs RL for drug discovery by searching retrosynthetic routes using Monte Carlo Tree Search (MCTS) in combination with an expansion policy network.

Several other works have explored training generative models on SMILES representations of molecules: CharacterVAE (Gómez-Bombarelli et al., 2016) is the first such model that is based on a VAE with recurrent encoder and decoder networks. GrammarVAE (Kusner et al., 2017) and SDVAE (Dai et al., 2018) constrain the decoding process to follow particular syntactic and semantic rules.

A related line of research considers training deep generative models to output graph-structured data directly. Several works explored auto-encoder architectures utilizing graph convolutions for link prediction within graphs (Kipf & Welling, 2016b; Grover et al., 2017; Davidson et al., 2018). Johnson (2017); Li et al. (2018b); You et al. (2018); Li et al. (2018a) on the other hand developed likelihood-based methods to directly output graphs of arbitrary size in a sequential manner. Several related works have explored extending VAEs to generate graphs directly, examples include the GraphVAE (Simonovsky & Komodakis, 2018), Junction Tree VAE (Jin et al., 2018) and the NeVAE (Samanta et al., 2018) model.

For link prediction within graphs, a range of adversarial methods have been introduced in the literature (Minervini et al., 2017; Wang et al., 2018; Bojchevski et al., 2018). This class of models, however, is not suitable to generate molecular graphs from scratch, which makes direct comparison infeasible.

5. Experiments

We compare MolGAN against recent neural network-based drug generation models in a range of experiments on established benchmarks using the QM9 (Ramakrishnan et al., 2014) chemical database. We first focus on studying the effect of the λ parameter to find the best trade-off between the GAN and RL objective (see Section 5.1). We then compare MolGAN with ORGAN (Guimaraes et al., 2017) since it is the most related work to ours: ORGAN is a sequential generative model operating on SMILES representations, optimizing towards several chemical properties with an RL objective (see Section 5.2). We also compare our model against variational autoencoding methods (Section 5.3) such as CharacterVAE (Gómez-Bombarelli et al., 2016), GrammarVAE (Kusner et al., 2017), as well as a recent graph-based generative model: GraphVAE (Simonovsky & Komodakis, 2018).

Dataset In all experiments, we used QM9 (Ramakrishnan et al., 2014) a subset of the massive 166.4 billion molecules GDB-17 chemical database (Ruddigkeit et al., 2012). QM9 contains 133,885 organic compounds up to 9 heavy atoms: carbon (C), oxygen (O), nitrogen (N) and fluorine (F).

Generator architecture The generator architecture is fixed for all experiments. We use $N = 9$ as the maximum number of nodes, $T = 5$ as the number of atom types (C, O, N, F, and one padding symbol), and $Y = 4$ as the number of bond types (single, double, triple and no bond). These dimensionalities are enough to cover all molecules in QM9. The generator takes a 32-dimensional vector sampled from a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and process it with a 3-layer MLP of $[128, 256, 512]$ hidden units respectively, with $\tanh$ as activation functions. Eventually, the last layer is linearly projected to match $\boldsymbol{X}$ and $\boldsymbol{A}$ dimensions and normalized in their last dimension with a $\mathrm{softmax}$ operation ($\mathrm{softmax}(\boldsymbol{x})_i = \exp(\boldsymbol{x_i}) / \sum_{i=1}^{D} \exp(\boldsymbol{x}_i)$).

Discriminator and reward network architecture Both networks use a RelationalGCN encoder (see Eq. 5) with two layers and $[64, 32]$ hidden units, respectively, to process the input graphs. Subsequently, we compute a 128-dimensional graph-level representation (see Eq. 6) further processed by a 2-layer MLP of dimensions $[128, 1]$ and with $\tanh$ as hidden layer activation function. In the reward network, we further use a sigmoid activation function on the output.

Evaluation measures We measure the following statistics as defined in Samanta et al. (2018): validity, novelty, and uniqueness. *Validity* is defined as the ratio between

the number of valid and all generated molecules. *Novelty* measures the ratio between the set of valid samples that are not in the dataset and the total number of valid samples. Finally, *uniqueness* is defined as the ratio between the number of unique samples and valid samples and it measures the degree of variety during sampling.

Training In all experiments, we use a batch size of 32 and train using the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 10^{-3}. For each setting, we employ a grid search over dropout rates $\in \{0.0, 0.1, 0.25\}$ (Srivastava et al., 2014) as well as over discretization variations (as described in Section 3.1). We always report the results of the best model depending on what we are optimizing for (e.g., when optimizing solubility we report the model with the highest solubility score – when no metric is optimized we report the model with the highest sum of individual scores). Although the use of WGAN should prevent, to some extent, undesired behaviors like *mode collapse* (Salimans et al., 2016), we notice that our models suffer from that problem. We leave addressing this issue for future work. As a simple countermeasure, we employ early stopping to avoid completely collapsed modes. In particular, we use the unique score to measure the degree of collapse of our models since it intrinsically indicates how much variety there is in the generation process. We set an arbitrary threshold of 2% under which we consider a model to be collapsed and stop training.

During early stages of our work, we noticed that the reward network needs several epochs of pre-training before being used to propagate the gradient to the generator, otherwise the generator easily diverges. We think this happens because at the beginning of the training, $\hat{R}_\psi$ does not predict the reward accurately and then it does not optimize the generator well. Therefore, in each experiment, we train the generator for the first half of the epochs without the RL component, but using the WGAN objective only. We train the reward network during these epochs, but no RL loss is used to train the generator. For the second half of the epochs we use the combined loss in Equation 4.

5.1. Effect of λ

As in Guimaraes et al. (2017), the λ hyperparameter regulates a trade-off between maximizing the desired objective and regulating the generator distribution to match the data distribution. We study the effects of $\lambda \in \{0.0, 0.01, 0.05, 0.25, 0.5, 0.75, 1.0\}$ on the solubility metric (see Section 5.2 for more details).

We train for 300 epochs on the 5k subset of QM9 used in Guimaraes et al. (2017). We use the best λ parameter – determined via the model with the maximum sum of valid, unique, novel, and solubility scores – on all other experiments (Section 5.2 and 5.3) without doing any further search.

Results We report results in Table 1. We observe a clear trend towards higher validity scores for lower values of λ. This is likely due to the implicit optimization of valid molecules since invalid ones receive zero reward during training. Therefore, if the RL loss component is strong, the generator is optimized to generate mostly valid molecular graphs. Conversely, it appears that λ does not mainly affect the unique and novel scores. Notice that these scores are not optimized, neither directly nor indirectly, and therefore they are a result of model architecture, hyperparameters, and training procedure. Indeed, the unique score is always close to 2% (which is our threshold) indicating that models appear to collapse (even in the RL only case) if we do not apply early stopping.

Since λ controls the trade-off between the WGAN and RL losses, it is not surprising that $\lambda = 0$ (i.e., only RL in the second half of training) results in the highest valid and solubility scores compared to other values. The λ value with the highest sum of scores is $\lambda = 0$. We use this value for subsequent experiments.

Algorithm	Valid	Unique	Novel	Solubility
$\lambda = 0$ (full RL)	**99.8**	2.3	97.9	**0.86**
$\lambda = 0.01$	98.2	2.2	**98.1**	0.74
$\lambda = 0.05$	92.2	2.7	95.0	0.67
$\lambda = 0.1$	87.3	**3.2**	87.2	0.56
$\lambda = 0.25$	88.2	2.1	88.2	0.65
$\lambda = 0.5$	86.6	2.1	87.5	0.48
$\lambda = 0.75$	89.6	2.8	89.6	0.57
$\lambda = 1$ (no RL)	87.7	2.9	97.7	0.54

Table 1. Comparison of different combinations of RL and GAN objectives on the small 5k dataset after GAN-based pretraining for 150 epochs. All values are reported in percentages except for the solubility score.

5.2. Objectives optimization

Similarly to the previous experiment, we train our model for 300 epochs on the 5k QM9 subset while optimizing the same objectives as Guimaraes et al. (2017) to compare against their work. Moreover, we also report results on the full dataset trained for 30 epochs (note that the full dataset is 20 times larger than the subset). We choose to optimize the following objectives which represent qualities typically desired for drug discovery:

Druglikeness: how likely a compound is to be a drug. The Quantitative Estimate of Druglikeness (QED) score

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	88.2	69.4*	9.63*	0.55	0.52	0.32	0.35
	OR(W)GAN	85.0	8.2*	10.06*	0.95	0.60	0.54	0.47
	Naive RL	97.1	54.0*	9.39*	0.80	0.57	0.53	0.50
	MolGAN	**99.9**	2.0	1.66	0.95	**0.61**	0.68	0.52
	MolGAN (QM9)	**100.0**	2.2	4.12	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	96.5	45.9*	8.66*	0.92	0.51	0.83	0.45
	OR(W)GAN	97.6	30.7*	9.60*	**1.00**	0.20	0.75	0.84
	Naive RL	97.7	13.6*	10.60*	0.96	0.52	0.83	0.46
	MolGAN	**99.4**	2.1	1.04	0.75	0.52	**0.90**	0.67
	MolGAN (QM9)	**100.0**	2.1	2.49	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	94.7	54.3*	8.65*	0.76	0.50	0.63	0.55
	OR(W)GAN	94.1	20.8*	9.21*	0.90	0.42	0.66	0.54
	Naive RL	92.7	100.0*	10.51*	0.75	0.49	0.70	0.78
	MolGAN	**99.8**	2.3	0.58	**0.97**	0.45	0.42	**0.86**
	MolGAN (QM9)	**99.8**	2.0	1.62	**0.99**	0.44	0.22	**0.89**
All/Alternated	ORGAN	96.1	97.2*	10.2*	0.92	**0.52**	0.71	0.53
All/Simultaneously	*MolGAN*	**97.4**	2.4	2.12	0.91	0.47	**0.84**	**0.65**
All/Simultaneously	*MolGAN (QM9)*	**98.0**	2.3	5.83	**0.93**	0.51	**0.82**	**0.69**

Table 2. Gray cells indicate directly optimized objectives. Baseline results are taken from Guimaraes et al. (2017) (Table 1) and * indicates results reproduced by us using the code provided by the authors.

quantifies compound quality with a weighted geometric mean of desirability scores capturing the underlying data distribution of several drug properties (Bickerton et al., 2012).

Solubility: the degree to which a molecule is hydrophilic. The log octanol-water partition coefficient (logP), is defined as the logarithm of the ratio of the concentrations between two solvents of a solute (Comer & Tam, 2001).

Synthetizability: this measure quantifies how easy a molecule is to synthesize. The Synthetic Accessibility score (Ertl & Schuffenhauer, 2009) is a method to estimate the ease of synthesis in a probabilistic way.

All scores are normalized to lie within $[0, 1]$. We assign a score of zero to invalid compounds (i.e., implicitly we are also optimizing a validity score). We also measure, without optimizing for it, a diversity score which indicates how likely a molecule is to be diverse with respect to the dataset. This measure compares sub-structures between samples and a random subset from the dataset indicating how many repetitions there are.

For evaluation, we report average scores for 6400 sampled compounds as in (Guimaraes et al., 2017). Additionally, we re-run experiments from (Guimaraes et al., 2017) to report unique scores and execution time since they are not provided in the original work. Differently from ORGAN, to optimize for all objectives, we do not alternate between optimizing them individually during training which in our case is not possible since the reward network is specific to a single type of reward. Thus, we instead optimize a joint reward which we define as the product (to lie within $\in [0, 1]$) of all objectives.

Results Results are reported in Table 2. Qualitative samples are provided in the Appendix (Figure 3). We observe that MolGAN models always converge to very high validity outputs $> 97\%$ at the end of the training. This is coherent as observed in the previous experiment, since also here there is an implicit optimization of validity. Moreover, in all single metrics settings, our models beat ORGAN models in terms of valid scores as well as all the three objective scores we optimize for.

We argue that this should be mainly due to two factors: i) intuitively, it should be easier to optimize a molecular graph predicted as a single sample than to optimize an RNN model that outputs a sequence of characters, and ii) using the deterministic policy gradient instead of REINFORCE effectively provides a better gradient and it improves the sampling procedure towards metrics while penalizing invalid graphs.

Training on the full QM9 dataset for 10 times fewer epochs further improves results in almost all scores. During training, our algorithm observes more different samples, and therefore it can learn well with much fewer iterations. Moreover, it can observe molecules with more diverse structures and properties.

As previously observed in Section 5.1, also in this experiment the unique score is always close to 2% confirming our hypothesis that our models are susceptible to *mode collapse*. This is not the case for the ORGAN baseline. Dur-

Algorithm	Valid	Unique	Novel
CharacterVAE	10.3	67.5	90.0
GrammarVAE	60.2	9.3	80.9
GraphVAE	55.7	**76.0**	61.6
GraphVAE/imp	56.2	42.0	75.8
GraphVAE NoGM	81.0	24.1	61.0
MolGAN	**98.1**	10.4	**94.2**

Table 3. Comparison with different algorithms on QM9. Values are reported in percentages. Baseline results are taken from Simonovsky & Komodakis (2018).

ing sampling, ORGAN generates sequences of maximum 51 characters which allows it to generate larger molecules whereas our model is (by choice) constrained to generate up to 9 atoms. This can explain the difference in unique score since the chance of generating different molecules in a smaller space is much lower. Notice that in ORGAN, the RL component relies on REINFORCE, and the unique score is optimized penalizing non-unique outputs which we do not.

In terms of training time, our model outperforms ORGAN by a large margin when training on the 5k dataset (at least $\sim$5 times faster in each setting), as we do not rely on sequential generation or discrimination. Both ORGAN and MolGAN have a comparable number of parameters, with the latter being approximately 20% larger.

5.3. VAE Baselines

In this experiment, we compare MolGAN against recent likelihood-based methods that utilize VAEs. We report a comparison with CharacterVAE (Gómez-Bombarelli et al., 2016), GrammarVAE (Kusner et al., 2017), and GraphVAE (Simonovsky & Komodakis, 2018). Here we train using the complete QM9 dataset. Naturally, we compare only with metrics that measure the quality of the generative process since the likelihood is not computed directly in MolGAN. Moreover, we do not optimize any particular chemical property except validity (i.e., we do not optimize any metric described above, but we optimize towards chemically valid compounds). The final evaluation scores are an average from 10^4 random samples. The number of samples differs from the previous experiment to be in line with the setting in Simonovsky & Komodakis (2018).

Results Results are reported in Table 3. Training on the full QM9 dataset (without optimizing any metric except validity) results in a model with a higher unique score compared to the ones in Section 5.2.

Though the unique score of MolGAN is slightly higher compared to GrammarVAE, the other baselines are superior in terms of this score. Even though here we do not consider our model to be collapsed, such a low score confirms our hypothesis that our model is prone to mode collapse. On the other hand, we observe significantly higher validity scores compared to the VAE-based baselines.

Notice that, differently from our approach, VAEs optimize the evidence lower bound (ELBO) and there is no explicit nor implicit optimization of output validity. Moreover, since a part of the ELBO maximizes reconstruction of the observations, the novelty in the sampling process is not expected to be high since it is not optimized. However, in all reported methods novelty is $> 60\%$ and, in the case of CharacterVAE, 90%. Though CharacterVAE can achieve a high novelty score, it underperforms in terms of validity. MolGAN, on the other hand, achieves both high validity and novelty scores.

6. Conclusions

In this work, we have introduced MolGAN: an implicit generative model for molecular graphs of small size. Through joint training with a GAN and an RL objective, our model is capable of generating molecular graphs with both higher validity and novelty than previous comparable VAE-based generative models, while not requiring a permutation-dependent likelihood function. Compared to a recent SMILES-based sequential GAN model for molecular generation, MolGAN can achieve higher chemical property scores (such as solubility) while allowing for at least $\sim$5x faster training time.

A central limitation of our current formulation of MolGANs is their susceptibility to mode collapse: both the GAN and the RL objective do not encourage generation of diverse and non-unique outputs whereby the model tends to be pulled towards a solution that only involves little sample variability. This ultimately results in the generation of only a handful of different molecules if training is not stopped early.

We think that this issue can be addressed in future work, for example via careful design of reward functions or some form of pre-training. The MolGAN framework taken together with established benchmark datasets for chemical synthesis offer a new test bed for improvements on GAN stability with respect to the issue of mode collapse. We believe that insights gained from such evaluations will be valuable to the community even outside of the scope of generating molecular graphs. Lastly, it will be promising to explore alternative generative architectures within the MolGAN framework, such as recurrent graph-based generative models (Johnson, 2017; Li et al., 2018b; You et al., 2018), as our current one-shot prediction of the adjacency tensor is most likely feasible only for graphs of small size.

Acknowledgements

The authors would like to thank Luca Falorsi, Tim R. Davidson, Herke van Hoof and Max Welling for helpful discussions and feedback. T.K. is supported by the SAP Innovation Center Network.

References

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

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

Bojchevski, Aleksandar, Shchur, Oleksandr, Zügner, Daniel, and Günnemann, Stephan. Netgan: Generating graphs via random walks. In *International Conference on Machine Learning*, 2018.

Bronstein, Michael M, Bruna, Joan, LeCun, Yann, Szlam, Arthur, and Vandergheynst, Pierre. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Bruna, Joan, Zaremba, Wojciech, Szlam, Arthur, and LeCun, Yann. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Comer, John and Tam, Kin. *Lipophilicity profiles: theory and measurement*. Wiley-VCH: Zürich, Switzerland, 2001.

Dai, Hanjun, Tian, Yingtao, Dai, Bo, Skiena, Steven, and Song, Le. Syntax-directed variational autoencoder for molecule generation. In *International Conference on Machine Learning*, 2018.

Davidson, Tim R, Falorsi, Luca, De Cao, Nicola, Kipf, Thomas, and Tomczak, Jakub M. Hyperspherical variational auto-encoders. In *UAI*, 2018.

Duvenaud, David K, Maclaurin, Dougal, Iparraguirre, Jorge, Bombarell, Rafael, Hirzel, Timothy, Aspuru-Guzik, Alán, and Adams, Ryan P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Ertl, Peter and Schuffenhauer, Ansgar. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

Gómez-Bombarelli, Rafael, Wei, Jennifer N, Duvenaud, David, Hernández-Lobato, José Miguel, Sánchez-Lengeling, Benjamín, Sheberla, Dennis, Aguilera-Iparraguirre, Jorge, Hirzel, Timothy D, Adams, Ryan P, and Aspuru-Guzik, Alán. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grover, Aditya, Zweig, Aaron, and Ermon, Stefano. Graphite: Iterative generative modeling of graphs. In *NIPS Bayesian Deep Learning Workshop*, 2017.

Guimaraes, Gabriel Lima, Sanchez-Lengeling, Benjamin, Farias, Pedro Luis Cunha, and Aspuru-Guzik, Alán. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Hamilton, William L, Ying, Rex, and Leskovec, Jure. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jin, Wengong, Barzilay, Regina, and Jaakkola, Tommi. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

Johnson, Daniel D. Learning graphical state transitions. *ICLR*, 2017.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Kipf, Thomas N and Welling, Max. Variational graph auto-encoders. In *NIPS Bayesian Deep Learning Workshop*, 2016b.

Kusner, Matt J, Paige, Brooks, and Hernández-Lobato, José Miguel. Grammar variational autoencoder. In *International Conference on Machine Learning*, pp. 1945–1954, 2017.

Li, Yibo, Zhang, Liangren, and Liu, Zhenming. Multi-objective de novo drug design with conditional graph generative model. *arXiv preprint arXiv:1801.07299*, 2018a.

Li, Yujia, Tarlow, Daniel, Brockschmidt, Marc, and Zemel, Richard. Gated graph sequence neural networks. *International Conference on Learning Representations*, 2016.

Li, Yujia, Vinyals, Oriol, Dyer, Chris, Pascanu, Razvan, and Battaglia, Peter. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018b.

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

Maddison, Chris J, Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Minervini, Pasquale, Demeester, Thomas, Rocktäschel, Tim, and Riedel, Sebastian. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*, 2017.

Ramakrishnan, Raghunathan, Dral, Pavlo O, Rupp, Matthias, and Von Lilienfeld, O Anatole. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.

Rezende, Danilo J., Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Ruddigkeit, Lars, Van Deursen, Ruud, Blum, Lorenz C, and Reymond, Jean-Louis. 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.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Samanta, Bidisha, De, Abir, Ganguly, Niloy, and Gomez-Rodriguez, Manuel. Designing random graph models using variational autoencoders with applications to chemical design. *arXiv preprint arXiv:1802.05283*, 2018.

Schlichtkrull, Michael, Kipf, Thomas N, Bloem, Peter, Berg, Rianne van den, Titov, Ivan, and Welling, Max. Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*, 2017.

Schneider, Gisbert and Fechner, Uli. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649, 2005.

Segler, Marwin HS, Preuss, Mike, and Waller, Mark P. Learning to plan chemical syntheses. *arXiv preprint arXiv:1708.04202*, 2017.

Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Simonovsky, Martin and Komodakis, Nikos. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

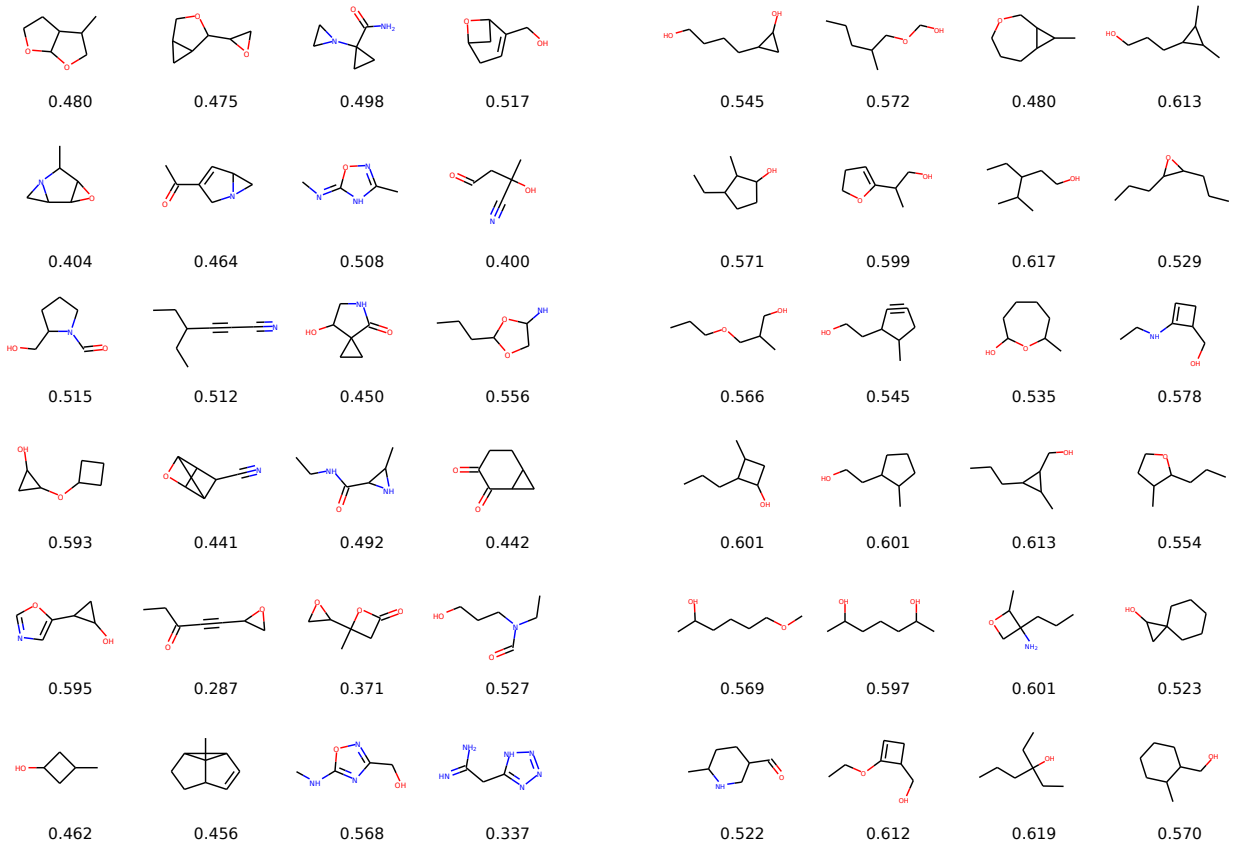
Wang, Hongwei, Wang, Jia, Wang, Jialin, Zhao, Miao, Zhang, Weinan, Zhang, Fuzheng, Xie, Xing, and Guo, Minyi. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*, 2018.

Weininger, David. 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.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

You, Jiaxuan, Ying, Rex, Ren, Xiang, Hamilton, William L, and Leskovec, Jure. Graphrnn: A deep generative model for graphs. In *International Conference on Machine Learning*, 2018.

Yu, Lantao, Zhang, Weinan, Wang, Jun, and Yu, Yong. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pp. 2852–2858, 2017.



(a) QM9 samples (b) MolGAN (QED) samples

Figure 3. Samples from the QM9 dataset (*left*) and MolGAN trained to optimize druglikeliness (QED) on the 5k QM9 subset (*right*). We also report their relative QED scores.