DiPol-GAN: Generating Molecular Graphs Adversarially with Relational Differentiable Pooling

Michael Guarino
Department of Computer Science
Marist College
Poughkeepsie, NY 12601
michael.guarino1@marist.edu

Alexander Shah
Department of Computer Science
Marist College
Poughkeepsie, NY 12601
alexander.shah2@marist.edu

Pablo Rivas
Department of Computer Science
Marist College
Poughkeepsie, NY 12601
Pablo.Rivas@Marist.edu

Abstract

Advances in deep generative modeling applied to irregular structures, such as graphs, have led to exciting advances specifically in the generation of graph structured data. This has been of particular importance to drug discovery as it directly applies to the problem of finding new molecular compounds. Many previous approaches to molecule generation have represented molecules using the SMILE (Simplified Molecular Input Line Entry System) strings format [1] rather than representing molecules directly as a graphs. Graph Neural Networks (GNNs) have shown state-of-the-art performance on many graph related tasks such as graph classification [2] and link prediction [3]. In this work we introduce DiPol-GAN, a generative adversarial network (GAN) approach to implicitly learning to generate molecular graphs. Using Differentiable Pooling, DiPol-GAN learns hierarchical representations of molecular graphs leading to more robust discriminator performance [4]. This work also proposes an extension of DIFFPOOL allowing it to handle graphs with multiple relation types such as different bond types that occur between atoms. To enhance the utility of this method we also constrain the learned latent representation with a reinforcement learning objective to shift the generation towards a targeted chemical property. Furthermore, we benchmark against other comparable models with similar claims. Preliminary results indicate that the proposed approach is competitive and for specific properties better than the benchmark.

1 Introduction

The extension of deep learning to graph data structures has become a popular research topic with compelling applications to the field of chemistry [5]. Of particular interest is the impact of deep generative models to *de novo* drug discovery. Drug discovery describes the process by which new candidate medications are found with the goal of identifying a new target molecule with specific desired properties. The key challenge presented is the vast size of the chemical space and the discrete nature of molecular structures [6]. The drug discovery process is expensive as even *de novo* approaches rely on some semblance of brute force. Improvements in methods to discover new drugs

Preprint. Under review.

with desired properties would have great impact. Our work presents a method to generate previously undiscovered molecular graphs with specific desired properties.

Generative models have been making astounding progress in the field of computer vision and natural language processing showing significant improvements in the quality of generated samples. Prominent approaches to generative modeling are autoregressive models [7], variational autoencoders (VAEs) [8], and generative adversarial networks (GANS) [9], each excelling in solving very specific types of problems. For the task of image synthesis autoregressive models generate high quality images but can be slow to evaluate and do not have a latent representation. VAEs train via maximum likelihood estimation making them more numerically stable; however, they can produce lower quality results [10]. GANs implicitly learn model parameters without having to specify a likelihood [11] and are known to produce higher quality results but are susceptible to generating lower diversity samples (a.k.a. mode collapse), and are prone to numerical instability making them difficult to train even with recent advances in adversarial training theory [12, 13, 14, 15].

Generative models have started to make their way into the domain of graphs with works such as [11, 16, 17, 18, 19, 20]; however, learning to generate graphs presents problems for gradient based learning due to the discrete nature of graphs because of their arbitrary connectivity [20]. Likelihood-based models for graph generation are known to be stable but require expensive approximate graph matching procedures as seen in GraphVAE [20] or require some fixed ordered representations of the graph as seen in Junction Tree VAE (JT-VAE) [18], which often is not feasible. Autoregressive models suchs as presented in GraphRNN [19] offer an interesting approach building a graph by conditionally modeling sequences of nodes where the sequence generators are jointly aware of each other; however, they lack a latent representation that would allow us to optimize for a desired chemical property.

GANs seem to present a clear advantage for the task of graph generation because they allow us to implicitly learn the data generation distribution without having to formalize a likelihood that necessitates a graph matching strategy as in the likelihood-based models (*e.g.* VAEs) while also allowing for the optimization of the learned distribution towards a desired chemical property.

In this work we present DiPol-GAN, summarized in Figure 1, which is a GAN-based approach to generating complete graph structures that are resistant to graph isomorphisms by learning to predict discrete connections. Through the use of a reinforcement learning objective DiPol-GAN's learned generation distribution is encouraged towards a desired chemical property. DiPol-GAN implements Differentiable Pooling (DIFFPOOL) [4] in its discriminator, which learns to aggregate nodes on the molecular graph in a hierarchical way improving classification accuracy and the quality of learned graph embeddings. We believe that through improving the discriminator's architecture it will encourage the generator to learn higher quality graph representations.

2 Related Work

Much existing work in small molecule drug discovery represent molecules using a SMILE string text representation derived from the molecule's actual graph [21, 22, 23, 24]. The work presented in this paper takes a different approach and deals directly with the molecular graph as it offers a richer representation, has greater flexibility, and asserts fewer prior assumptions. Models using this representation are essentially learning SMILE string syntax and hoping to learn something in which they could decode into a valid molecule. SMILES do not capture structural similarity between molecules and one molecule can have multiple smile representations. In the next paragraphs we review similar methodologies exposing where they fall short motivating us to make improvements.

2.1 Likelihood-based approaches

Simonovsky proposes GraphVAE [20], a likelihood based approach that outputs a probabilistic fully-connected graph, which then uses approximate graph matching to align the the generated graph with the ground truth in order to formulate its likelihood. The molecular graph is embedded into a latent space via a stochastic graph encoder and is then decoded by a graph decoder into a fully probabilistic graph. The expensive graphing matching algorithm, which is required for GraphVAE to forumlate a likelihood, prevents it from scaling to generate large graph structures. JT-VAE addresses the generation of a molecular graph by formulating a VAE and decomposing the molecular graph into junction trees where each substructures on the molecular graph is then encoded into latent

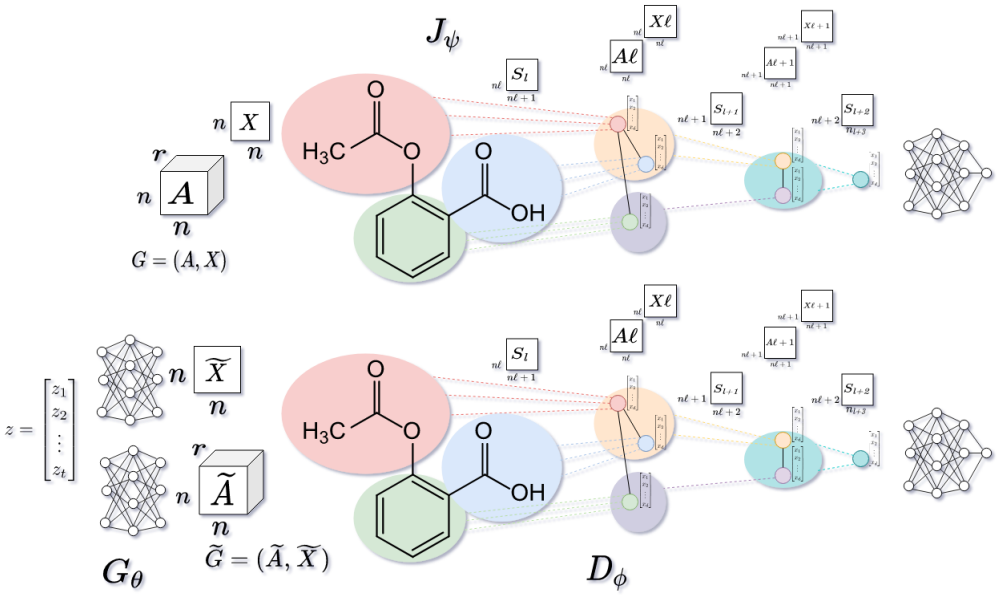

Figure 1: Proposed model architecture. J_ψ is a policy network, G_θ is the generator, and D_ϕ is the discriminator in a graph-based generative adversarial network. The generator is a graph $G = (\mathbf{A}, \mathbf{X})$ where $\mathbf{A}$ is the adjacency matrix, and $\mathbf{X}$ is the features matrix. The model implements a reinforcement learning reward mechanism.

space while the original molecular graph is also encoded into its respective molecular graph [18]. These encodings of the molecular graph are then decoded from the latent space into a learned graph. This approach is intriguing because of its exploitation of the molecule's substructure to allow for hierarchical encoding of the molecular graph. This approach does not provide a clear way to optimize for molecular properties in the latent space and is therefore not ideal for generating molecular graphs with specific properties.

2.2 Implicit (Adversarial) approaches

The major adversarial approaches to molecular graph generation are MolGAN [11] and Graph Convolutional Policy Networks (GCPNs) [17]. The formulation of MolGAN is most similar to the method proposed in this work. MolGAN implicitly learns to generate complete fixed size molecular graphs by inferring model parameters without having to specify a likelihood. This learned generative distribution is then further optimized towards specific chemical properties by way of a deterministic policy gradient algorithm. MolGAN implements the WGAN objective [13]; however, still suffers from mode collapse which impacts the diversity of generated samples [12]. GCPN has a dramatically different as it tackles molecular graph generation by formalizing the iterative construction of bonds and molecular scaffolds by a reinforcement learning agent using a policy network to predict the next action taken on the graph from a distribution of actions to maximize an overall reward, which is to generate a molecular graphs with a desired property. GCPN uses a generative adversarial network to formalize an adversarial reward that is used to ensure that the learned generative distribution resembles the data generations distribution.

3 Methods

3.1 Background

3.1.1 Generative adversarial network

GANs consist of two networks, a generator G_θ and a discriminator D_ϕ. The discriminator that learns to identify whether an example is from the model distribution or the data distribution while the generator learns to map from a prior distribution to a distribution resembling the data distribution [9]. In this way the generator and discriminator have competing objectives and thus can be seen as the following minimax objective:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \left[\log D_\phi(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log \left(1 - D_\phi \left(G_\theta(\mathbf{z}) \right) \right) \right], \tag{1}$$

where $\mathbf{x}$ is a multi-dimensional one-hot encoded vector representing the type of atom, and $\mathbf{z}$ is a sampled vector from an estimated prior distribution. The end result is a generator that implicitly learns to model the data distribution in order to improve its performance against the discriminator.

3.1.2 Molecular Graphs

We represent molecules as undirected graphs G containing a set of vertices V and edges E, the atoms and bonds on the graph, respectively; therefore $G = (V, E)$. Each vertex on the graph $v_i \in V$ contains an atom type d. Atoms on the graph relate to each other with atomic bonds as $(v_i, v_j) \in E$. These bond types are represented with $r \in \{1, \ldots, \mathbb{R}\}$. In this project we represent a molecular graph with two tensors $\mathbf{A}$ and $\mathbf{X}$. $\mathbf{A}$ is an adjacency matrix that describes which nodes are related to each other and the type of relation. $\mathbf{A} \in \mathbb{R}^{n \times n \times r}$ where n is a node on the graph. The feature tensor $\mathbf{X}$ contains node feature representations of size $\mathbf{X} \in \mathbb{R}^{n \times d}$. Both atom and bond information is one hot encoded.

3.2 Generator

Molecular graphs require more careful consideration of the relationships between nodes. Atoms can only form certain types of bonds with each other under specific conditions therefore our generator must have the capacity to appropriately model these relations. In order to model molecular graphs our generator samples from a latent space $\mathbf{z}$, uniformly initialized between (0,1), then outputs $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{X}}$ after being sent through a several affine transformations resulting in a continuous dense tensors. In obtain to obtain a molecular representation from $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{X}}$ we must be obtain a discrete representations of each atom type d and bond type r. An elegant way to obtain discrete representations, as suggested in [11, 20], is to use Gumbel-Softmax as it is also differentiable [25]. In order to ensure that we are modeling molecular bonding information we use Relational-GCN (RGCN) [26], a graph convolutional network that supports multiple relation types ensuring that edge information contributes to the learned graph embedding. The message passing function can be defined as follows:

$$\mathbf{h}_i^{\prime(\ell+1)} = f_s^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{x}_i) + \sum_{j=1}^{N} \sum_{y=1}^{Y} \frac{\tilde{\mathbf{A}}_{ijy}}{|\mathcal{N}_i|} f_y^{(\ell)}(\mathbf{h}_j^{(\ell)}, \mathbf{x}_i), \tag{2}$$

$$\mathbf{h}_i^{(\ell+1)} = \tanh(\mathbf{h}_i^{\prime(\ell+1)}) \tag{3}$$

where $\mathbf{h}_i^{(\ell)}$ is the signal from the center node i in neighborhood $\mathcal{N}_i$ at layer l in the network, $f_y^{(\ell)}$ is a linear transformation between layers of the network, $f_s^{(\ell)}$ is an affine function, and $1/\left|\mathcal{N}_i\right|$ is used to normalize the contribution of each neighborhood in case neighborhood size is not uniform which could affect the scale of the corresponding activations.

An RGCN serves a strategic purpose in the generator, since very unstable GAN training is often correlated with imbalances in the representational capacity between the generator and the discriminator.

3.3 Discriminator

The discriminator implements DIFFPOOL which allows information to be aggregated accross the graph in a hierarchical way by learning differentiable soft cluster assignments for nodes at a given

layer and maps them a set of clusters that are taken as input to the next layer GNN [4]. This method claims to improve graph classification accuracy by up to 10%. DIFFPOOL can be implemented using any GNN that adheres to the MPNN framework. DIFFPOOL expects graphs G as $(\mathbf{A}, \mathbf{F})$ with $\mathbf{A} \in \{0,1\}^{n \times n}$ as its adjacency matrix and $\mathbf{F} \in \mathbb{R}^{n \times d}$ as its feature matrix where n and d are nodes and features, respectively. The binary adjacency matrix, $\mathbf{A}$, purposed by DIFFPOOL must be extended in order to be appropriate for molecular graphs where bonds are inherently multinomial variables. We accomplish this by using RGCN modules as the first layer to DIFFPOOL. This allows for node and edge information to be considered. After the first layer of DIFFPOOL, adjacency is modeled as Bernouli variables which is appropriate because the molecular graph is no longer a molecular graph but rather a embedded representation.

The following equation shows a general "message passing" GNN architecture formalized by MPNN [5]:

$$\mathbf{H}^{(k)} = M(\mathbf{A}, \mathbf{H}^{(k-1)}; \boldsymbol{\theta}^{(k)}) \tag{4}$$

where $\mathbf{H}^{(k)} \in \mathbb{R}^{n \times d}$ are the node embeddings computed after k steps of message passing; $\boldsymbol{\theta}^{(k)}$ is a learned parameter; $\mathbf{H}^{(k-1)}$ are the node embeddings computed in the previous step; M is the message passing function

The following equation aggregates node embeddings $\mathbf{Z}^{(l)}$ and aggregates them according to the learned cluster assignment matrix $\mathbf{S}^{(l)}$ producing node embeddings for layer n_{l+1}:

$$\mathbf{X}^{(l+1)} = {\mathbf{S}^{(l)}}^T \mathbf{Z}^{(l)} \in \mathbb{R}^{n_{l+1} \times d} \tag{5}$$

The next equation shows the generated coarsened adjacency matrix $\mathbf{A}^{(l)}$ which models connectivity among pairs of nodes in the clusters at layer $l+1$:

$$\mathbf{A}^{(l+1)} = {\mathbf{S}^{(l)}}^T \mathbf{A}^{(l)} \mathbf{S}^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}} \tag{6}$$

Then, the following equation takes as input the adjacency matrix from Eq. (6) and generates a new embedding matrix for clusters at layer l:

$$\mathbf{Z}^{(l)} = \text{GNN}_{l,embed}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)}) \tag{7}$$

The next equation shows the learned cluster assignment matrix $\mathbf{S}^{(l)} \in \mathbb{R}^{n_l \times \pi_{l+1}}$ that map nodes from layer l to node clusters at layer $l+1$ by using the softmax activation function to classify connection between nodes at layer l and clusters at layer $l+1$:

$$\mathbf{S}^{(l)} = \text{softmax}\big(\text{GNN}_{l,pool}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})\big) \tag{8}$$

The auxiliary link prediction objective that is minimized at each layer and works to encourage neighboring nodes to pool together can be formally denoted as:

$$L_{LP} = ||\mathbf{A}^{(l)}, \mathbf{S}^{(l)} {\mathbf{S}^{(l)}}^T||_F, \tag{9}$$

where $||\cdot||_F$ is the Frobenius norm.

To regularize the entropy of the cluster assignment we minimize the following loss:

$$L_E = \frac{1}{n} \sum_{i=1}^{n} H(\mathbf{S}_i), \tag{10}$$

where H denotes the entropy function, and $\mathbf{S}_i$ is the i-th row of $\mathbf{S}$.

We use (7) and (8), *i.e.*, the base case layer 0, to accept an input adjacency $\mathbf{A}$ and feature $\mathbf{F}$ matrices obtain an embedding $\mathbf{Z}^l$ and and learned cluster assignment $\mathbf{S}^l$ that is used to aggregate node embeddings and relations with (5) and (6) respectively. DIFFPOOL also ensures permutation invariance as long as the GNN module used in (4) is also permutation invariant [5].

3.4 Policy Network

DiPol-GAN introduces a third network for the purpose of directing the generator's learned distribution towards a desired property. This network's architecture is identical to the original discriminator's that implements DIFFPOOL. The policy network samples over a distribution of policies in order to maximize performance on the reward function. Policies are represented as a parametric function $\theta, \pi_\theta(a|s)$ to maximize performance on the reward (objective) function. Our reward function is formalized as a the mean squared error between the calculated desired chemical property of our generated graph and the desired property. Gradients are then propagated back to the generator encouraging it to learn the a distribution that maximizes this reward.

4 Evaluation

4.1 QM9 Dataset

The QM9 dataset [27] contains about 134,000 stable small organic molecules consisting of Carbon, Hydrogen, Oxygen, Nitrogen, and Flourine. Therefore QM9 consists of 4 distinct atomic numbers, 4 bond types, and each molecules is up to 9 heavy atoms. QM9 is a popular benchmark for machine learning in the chemistry community [27]. We reserve 10 percent of the data for test, another 10 percent for validation, and the remain 80 percent of the data is used for training.

4.2 Quality Metrics

The criteria we are using to determine the performance of our generative model requires us to define V all valid molecules generated from G while C is the list of all chemically valid molecules. The $Validity$ of generated molecules is measured by its adherence to known constraints of chemical structures and is formally defined as $Valid = |V| / n$ where n is the sample size for a given batch. $Uniqueness$ is the fraction of unique valid graphs formalized as $Unique = |\text{set}(C))| / |(C)|$. $Novel$ is the novel out of the entire dataset $Novel = 1 - |\text{set}(C)) \cap \text{QM}\,9| / |\text{set}(C))|$ [20]. We are also using a reinforcement learning objective in order to optimize the generation toward a specific desired property and therefore also consider that part of the performance criteria of the model. We are specifically interested in $\log P$ solubility and a drug candidate score.

5 Experiments

GANs are susceptible is several undesirable properties that are unfavorable for the domain of property specific molecular graph generation. Numerical instability which is marked by severe oscillation of model parameters preventing convergence, mode collapse, significant imbalance between the generator and discriminator. Of most concern is mode collapse, which occurs when the generator gets stuck in a local minimum and generates low variety samples. As GAN training theory is evolving we try several proposed methods in order achieve favorable results for molecular graph generation.

We start by investigating methods used in MolGAN. First using the improved WGAN [28] loss function using the Wasserstein distance, and the Earth Mover distance since it has been shown to improve GAN training performance because even in low dimensional manifolds the Wasserstein disantance still maintains a smooth shape and therefore improves numerical stability using gradient based training methods. The Wasserstein distance, D_W, is given by:

$$D_W[p||q] = \frac{1}{K} \sup_{||f||_L < K} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[f(\mathbf{x})]. \tag{11}$$

We also experimented with the gradient penalty alternative introduced by [13] which has improved on the practice of gradient clipping. The following gradient penalty is only used when training the discriminator:

$$L(\mathbf{x}^{(i)}, G_\theta(\mathbf{z}^{(i)}); \phi) = \underbrace{-D_\phi(\mathbf{x}^{(i)}) + D_\phi(G_\theta(\mathbf{z}^{(i)}))}_{\text{WGAN loss}} + \underbrace{\alpha\big(|| \bigtriangledown_{\widehat{\mathbf{x}}^{(i)}} D_\phi(\widehat{\mathbf{x}}^{(i)})|| - 1\big)^2}_{\text{gradient penalty}} \tag{12}$$

where α is a hyperparameter, $\widehat{\mathbf{x}}^{(i)}$ is sampled linear combination between $\mathbf{x}^{(i)} \sim p_{\text{data}}(\mathbf{x})$, $G_\theta(\mathbf{z}^{(i)})$, $\mathbf{z}^{(i)} \sim p_{\mathbf{z}}(\mathbf{z})$, and $\widehat{\mathbf{x}}^{(i)} = \epsilon\mathbf{x}^{(i)} + (1-\epsilon)G_\theta(\mathbf{z}^{(i)})$ with $\epsilon \sim \mathcal{U}(0,1)$.

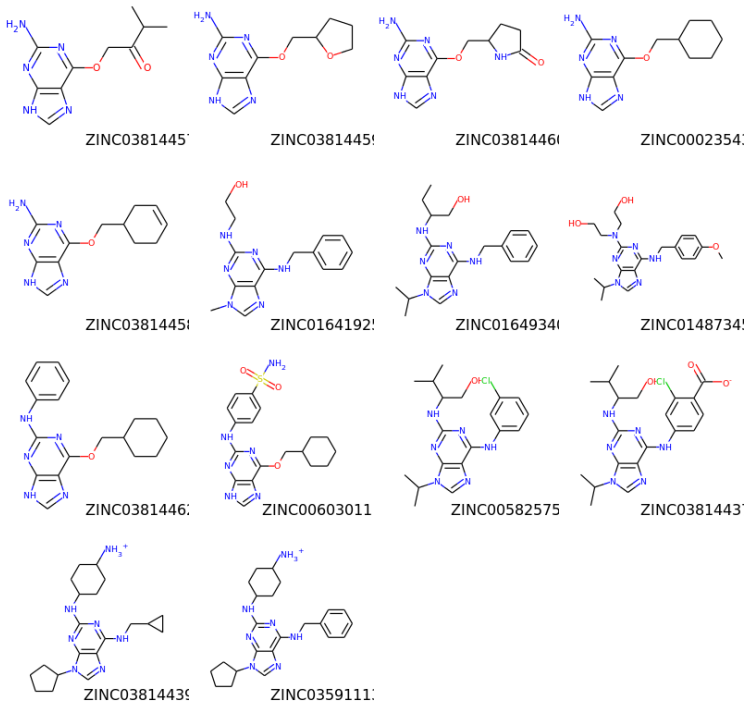

Figure 2: Visualized 2D representation of small molecules generated by DiPol-GAN while training on QM9. List **z** space and architecture specs.

The key intuition is that we stack L GNN modules and learn to assign nodes to clusters at layer l in an end-to-end fashion, using embeddings generated from a GNN at layer $l-1$. Thus, we are using GNNs to both extract node embeddings that are useful for graph classification, as well to extract node embeddings that are useful for hierarchical pooling. The above method achieved good results but we were still experiencing the problem of mode collapse reported with MolGAN.

5.1 Results

Table 1: Comparisons with related molecular graph generation work, presented are test results.

Algorithm	Valid (%)	Uniqueness	Diversity	Druglikeliness	Time (s)
MolGAN	77.5 $\pm$42.0	0.0977 $\pm$0.157	0.877 $\pm$0.214	0.513 $\pm$0.221	16473 $\pm$651
DiPo (WGAN)	100 $\pm$ 0.0	0.883 $\pm$ 0.013	0.9987 $\pm$ 0.45	0.648 $\pm$ 0.213	19484 $\pm$ 134

Using the QM9 dataset for the task of goal directed molecular graph generation can be problematic in that there are 134k molecular graph structures lacking software to calculate chemical properties on a GPU in a timely manner. Nearly all related work report lengthy training times with GraphVAE reporting $>$ 5 hours to complete one training epoch. It is for this reason that DiPolGAN presents qualitative metrics on the validation data set for this submission. In our work, training on an NVIDIA 2080Ti DiPOLGAN took 19484 ± 134 seconds per training step on the QM9 dataset. The validation metric reported was achieved after 10 training epochs.

6 Conclusion

DiPOLGAN presents a method for generating molecular graphs with specific chemical properties via hierarchical differentiable pooling to improve discriminator performance while also optimizing towards a a reinforcement learning objective via the policy network. Through learning hierarchical

graph representations it is our intuition from representation learning that our discriminator and policy networks are learning to hierarchically filter learned representation towards information present to improve each specific network's objective and therefore implicitly encourage the generation of realistic molecular graphs.

References

[1] 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.

[2] Zhang Xinyi and Lihui Chen. Capsule graph neural network. In *International Conference on Learning Representations*, 2019.

[3] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

[4] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[6] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[7] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

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

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

[10] 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.

[11] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[12] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[14] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[15] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[16] Bidisha Samanta, Abir De, Gourhari Jana, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs. *arXiv preprint arXiv:1802.05283*, 2018.

[17] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pages 6410–6421, 2018.

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

[19] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

[20] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[21] 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.

[22] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954. JMLR. org, 2017.

[23] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

[24] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

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

[26] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[27] 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.

[28] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.