MolGAN: An implicit generative model for small molecular graphs

Authors: Nicola De Cao, Thomas Kipf

Presenters: Daisy Zheng, Nilay Shah, and Nima Zaghari

Slides adapted from N. De Cao and T. Kipf ICML Deep Generative Models Workshop (2018)

Overview

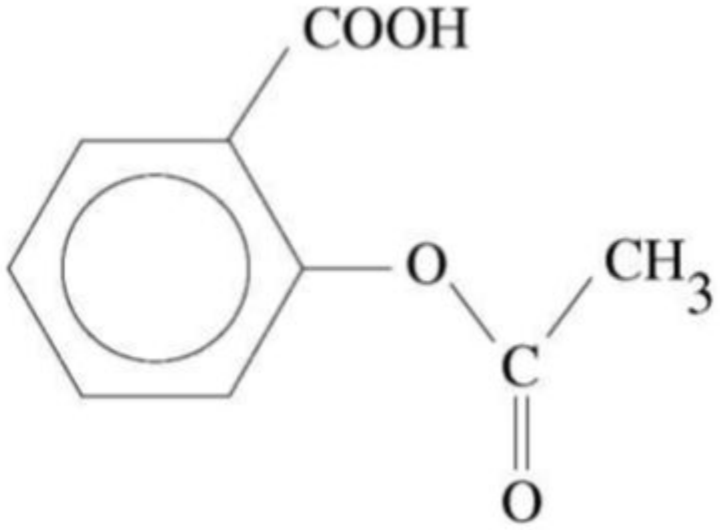
MolGAN incorporates the following:

- Automatic generation of small drug-like molecules
- Generative Adversarial Net, Graph Neural Network, and Reinforcement Learning
- Optimization of biochemical properties such as solubility
- Provides a pathway for in-silico screening in ML

Drug Background

Drug Properties

- Useful bioactivity
- Controllable side effect
- Synthesizability
- After metabolism effect?

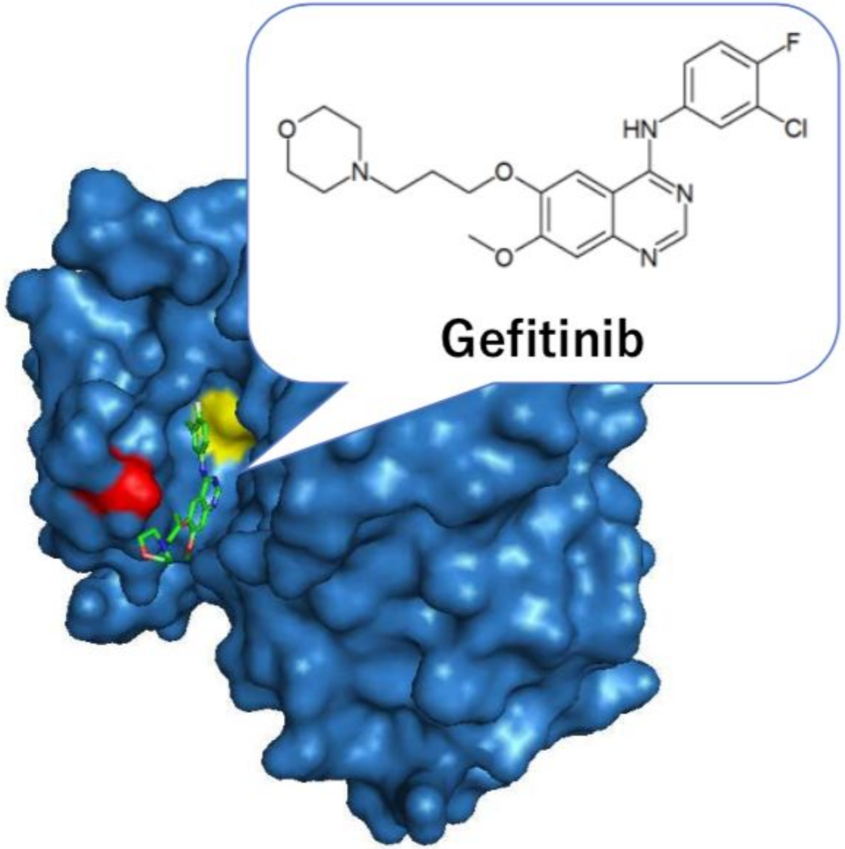


In-silico screening provides an alternative to expensive animal/human experiments

Screening Simulation

For every target drug we must:

- Determine structure of target protein
- Make a decision on the target site
- Static affinity prediction
- Dynamic binding simulation



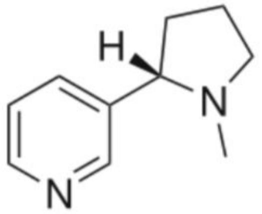
Mutated EGFR
(non small cell lung cancer)

What makes drug design difficult?

1. Large search space
    a. For 10 C/N/O atoms, there are over 60,000 permutations
    b. Only a small set of atomic permutations give a valid structure
2. Discrete Optimization of molecular structure
    a. Continuous optimization is not possible
3. Slight change in structure results in large effects
    a. COH and COOH are different

What makes drug design difficult?

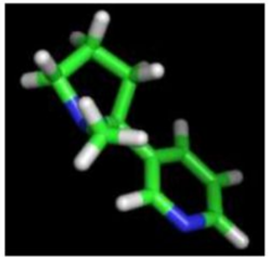
4. No appropriate data structure for molecular structure



Image

CN1CCC[C@H]1c2cccnc2

SMILES representation



3D structure (important for proteins)

5. Predicting biochemical properties is difficult

Can we solve this problem with ML?

1. Large search space
   a. Generative models, like GAN, can represent complex/high-dimensional data
2. Discrete optimization of molecular structure
   a. This paper will just do a screening and not do any fine-tuning for specific drugs
3. Slight change in structure results in large effects
   a. Affinity prediction is still difficult. ML is better for predicting general properties like solubility
4. No appropriate data structure for molecular structure
   a. Graph representation and GCNN
5. Predicting biochemical properties is difficult
   a. Can’t truly solve this. We would need to improve simulations.

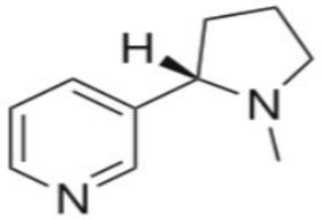
Background

Molecular Structure Representation

Image: Easily interpretable, but not efficient

SMILES: Rich information, but syntax is too strict

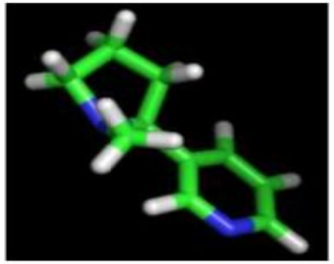
3D: Very rich information, large data size, invariance problem



2D Image

CN1CCC[C@H]1c2cccnc2

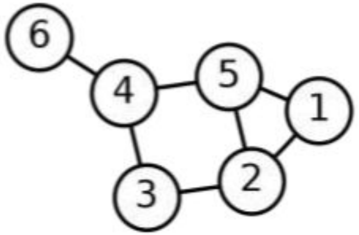
SMILES



3D structure

Molecules as Graphs

Graph : Network structure consist of nodes V and edges E



simple graph

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Adjacency matrix

Node = atom / Edge = bond → Graph = molecule

Node matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T \in \mathbb{R}^{N \times T}$ Adjacency tensor $\mathbf{A} \in \mathbb{R}^{N \times N \times Y}$

Implicit vs. likelihood-based methods

Likelihood-based methods:

- These methods, such as VAE, allow for easier and more stable optimization when compared to implicit generative models like GAN
- For graph-structured data, we want to be invariant to node ordering
  - This means we must do expensive graph matching or evaluate the likelihood explicitly for each permutation

Implicit methods:

- Implicit generative models, like the GAN framework, avoid the need for an explicit likelihood
- The discriminator of the GAN can be made invariant to node ordering
- The generator is free to choose any ordering

Generative Adversarial Networks

- Implicit generative models that allow for inference of model parameters without specifying a likelihood.
- GAN has 2 components:
  - Generative model G_θ, that learns a map from a prior to the data distribution to sample new data-points
  - Discriminative model D_φ, that learns to classify whether samples came from the data distribution rather than from G_θ.

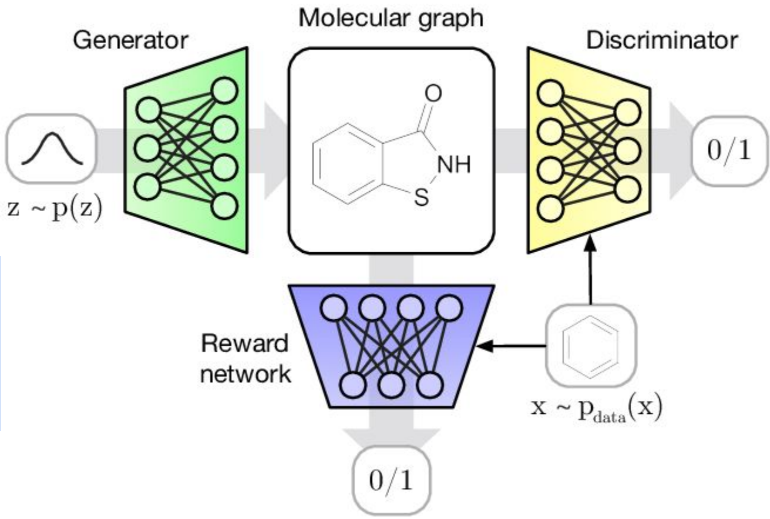
$$\min_{\theta} \max_{\phi} \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D_\phi(G_\theta(\boldsymbol{z})))]$$

Improved WGAN

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

Hyperparameter = 10

A sampled linear combination between $x^{(i)} \sim p_{data}(x)$ and $G_\theta(z^{(i)})$ with $z^{(i)} \sim p_z(z)$



Reinforcement Learning

Learning framework for robot movement

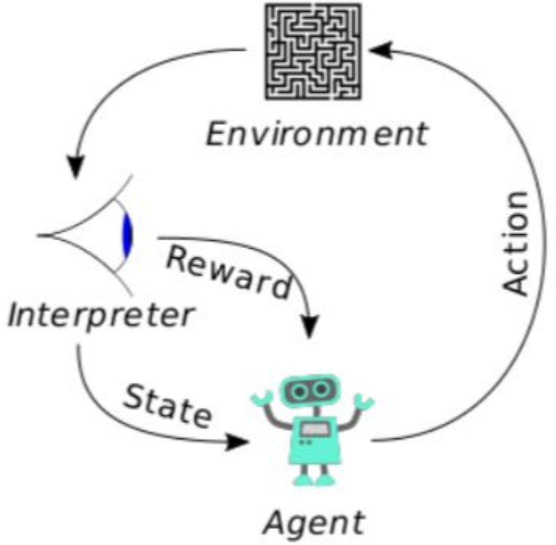
Action under an environment gives

a reward reflecting the goodness

ex) going toward a hole results in death of Mario

Optimizing the policy to maximize the reward

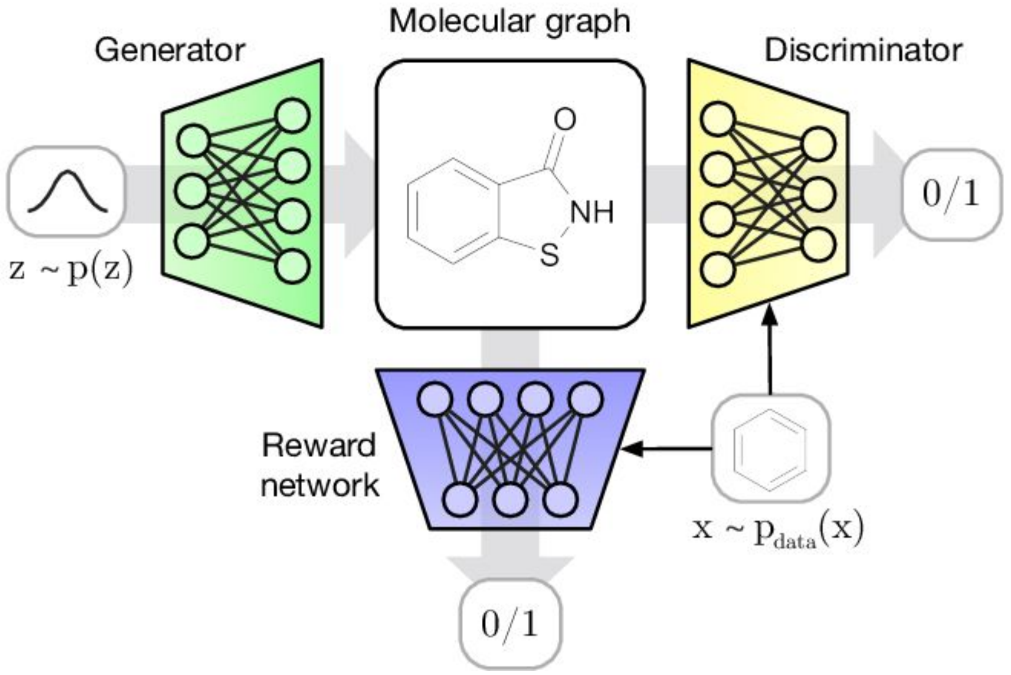
ex) Jump when a hole is located in front of Mario



Deterministic Policy Gradients

- In reinforcement learning, a stochastic policy is represented by $\pi_\theta(s) = p_\theta(a|s)$
    - 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.
- This paper chooses a deterministic policy gradient algorithm

Model Overview

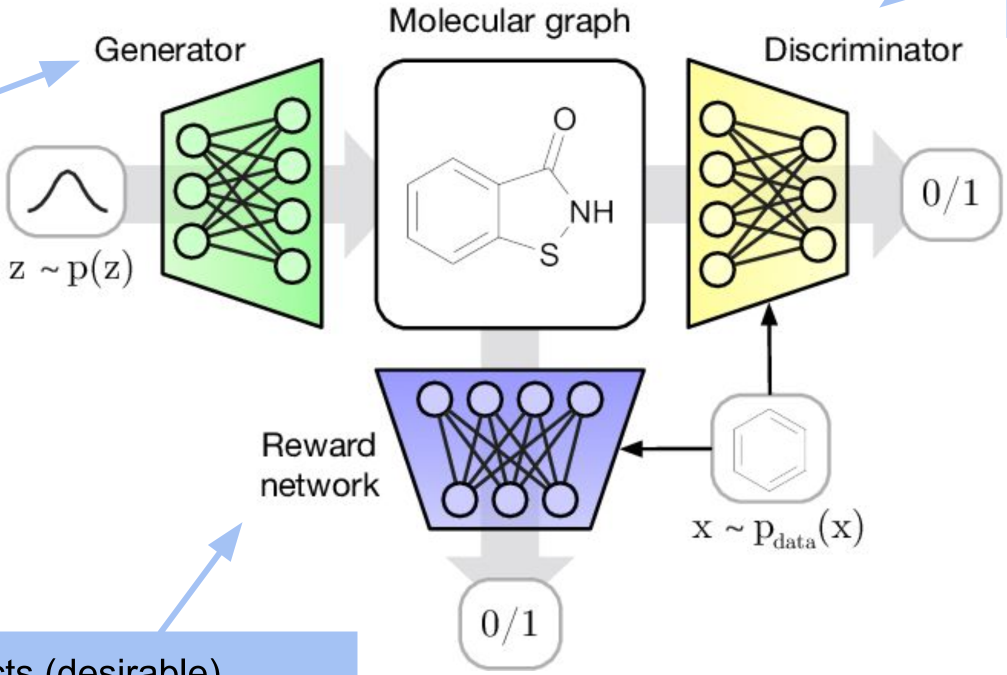


Model Overview

Transforms noise into a molecular structure

Distinguishes between molecules sampled from the generator and the dataset

Predicts (desirable) properties of the molecule



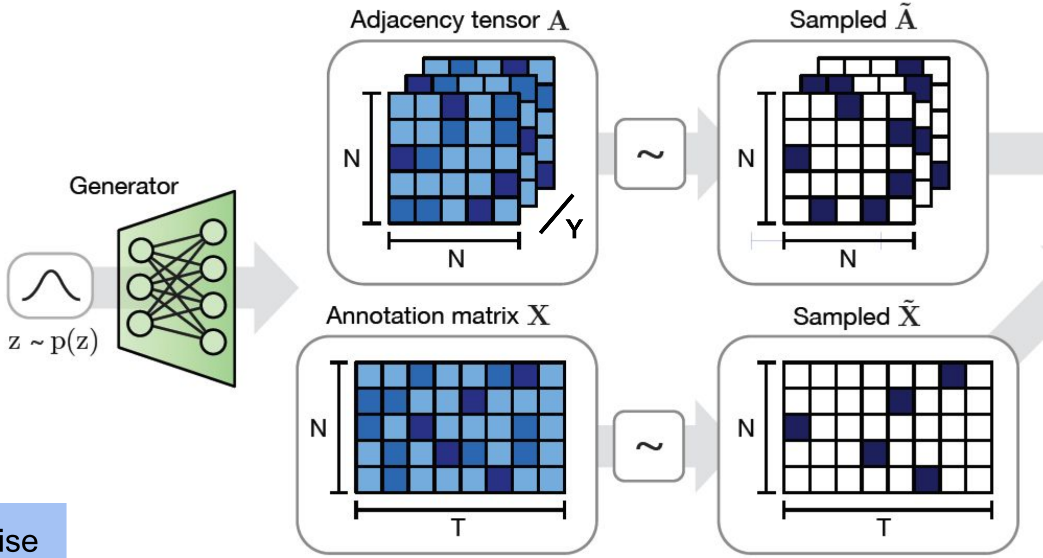
Generator

Multi-Layer Perceptron (MLP)

Predicts the entire graph at once

$G_\phi(z)$

D-dimensional vector representing random noise
$z \sim \mathcal{N}(0, \boldsymbol{I})$



Generator
$z \sim p(z)$
Adjacency tensor $\mathbf{A}$
N
N
Y
~
Sampled $\tilde{\mathbf{A}}$
N
N
Annotation matrix $\mathbf{X}$
N
T
~
Sampled $\tilde{\mathbf{X}}$
N
T

Outputs: X and A

- Dense continuous matrices
- Contains probabilities of categorical distributions over node (atom) and edge (bond) types

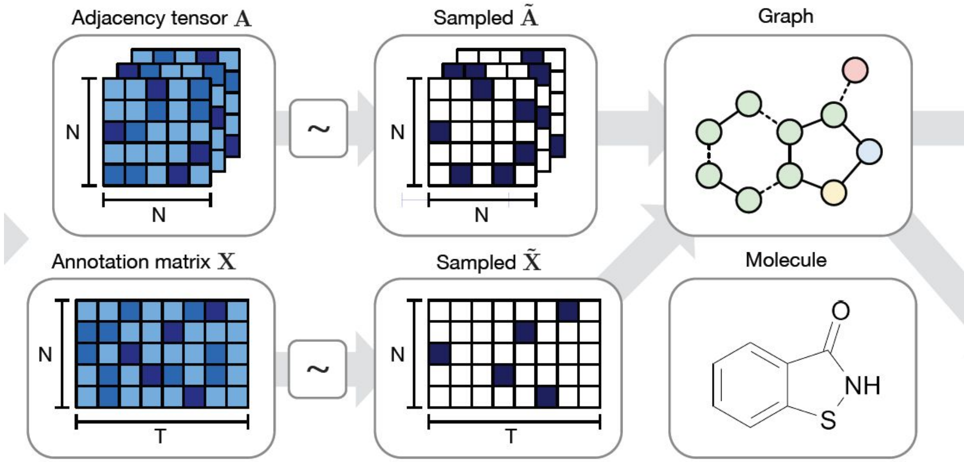
Generating the Molecule

Sample $\tilde{X}$ and $\tilde{A}$

- Sparse, discrete matrices
- Chooses an atom type and bond type for each node/edge

x_i : one-hot vector indicating node i's atom type

A_{ij} : one-hot vector indicating bond type between nodes i and j



Adjacency tensor $\mathbf{A}$
N
N
~
Sampled $\tilde{\mathbf{A}}$
N
N
Graph
Annotation matrix $\mathbf{X}$
N
T
~
Sampled $\tilde{\mathbf{X}}$
N
T
Molecule
O
NH
S

Training the Generator

Learns a transformation (mapping) from a prior distribution to the data distribution

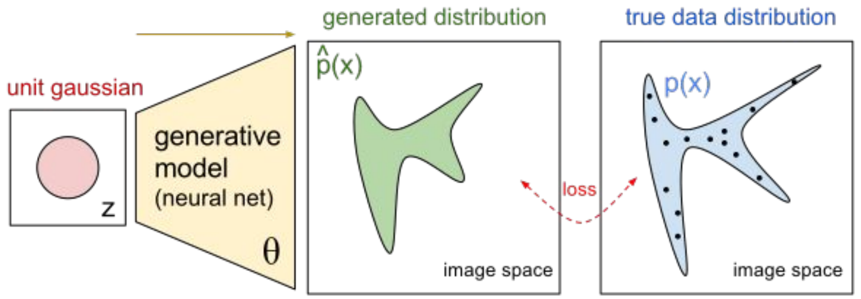
- Want generated samples to resemble data samples

Loss function:

- Linear combination of loss from WGAN objective and reinforcement learning

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

hyperparameter regulating trade-off between WGAN and RL



https://openai.com/blog/generative-models/

Discriminator

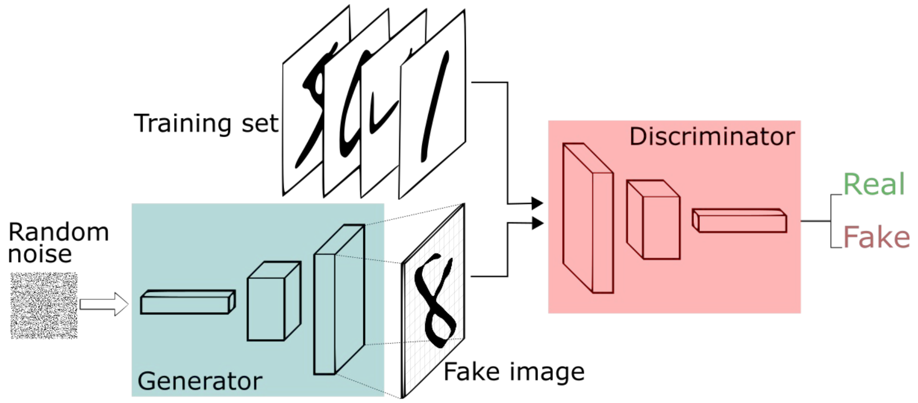
D_ϕ Input: graph
Output: scalar ∈ (-∞, +∞)

Takes samples from both the generator and the dataset

Learns to distinguish between:

- Generator samples (fake data)
- Dataset samples (real data)

Trained using WGAN objective



Reward Network

$\hat{R}_\psi$ Input: graph
Output: scalar ∈ (0, 1)

Takes samples from both the generator and the dataset

Approximates a reward function

- Trained to match scores provided by some external software (ex. RDKit)
    - Mean squared error objective
- Reinforcement learning feeds back the reward to the generator to optimize molecules towards non-differentiable metrics that measure useful properties

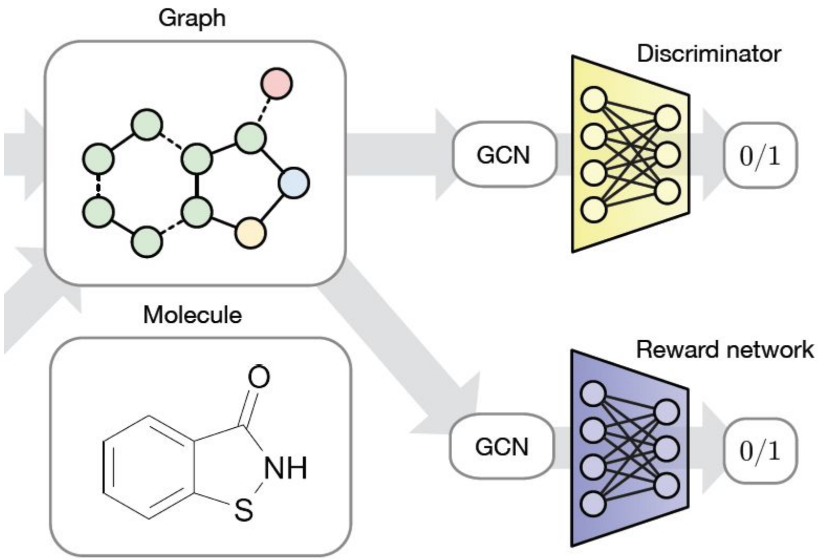
No reward (zero-score) assigned to invalid molecules

Discriminator + Reward Network

Same architecture: Graph Convolution Network (GCN)

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

$$\boldsymbol{h}_i^{(\ell+1)} = \tanh(\boldsymbol{h}_i'^{(\ell+1)})$$



Discriminator + Reward Network

Same architecture: Graph Convolution Network

linear transformation function

signal of node i at layer l

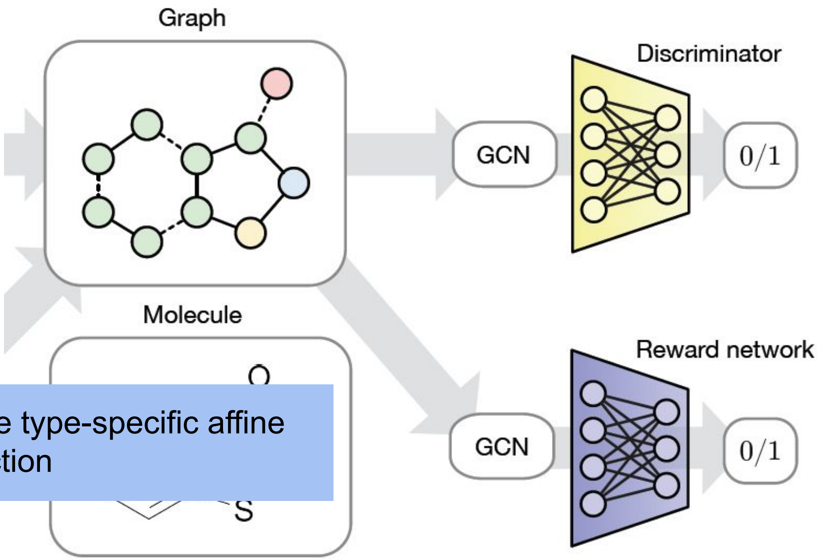
$$\boldsymbol{h}_i'^{(\ell+1)} = f_s^{(\ell)}(\boldsymbol{h}_i^{(\ell)}, \boldsymbol{x}_i) + \sum_{j=1}^{N}\sum_{y=1}^{Y} \frac{\tilde{\boldsymbol{A}}_{ijy}}{|\mathcal{N}_i|} f_y^{(\ell)}(\boldsymbol{h}_j^{(\ell)}, \boldsymbol{x}_i)$$

$$\boldsymbol{h}_i^{(\ell+1)} = \tanh(\boldsymbol{h}_i'^{(\ell+1)})$$

activation function

normalization factor $1/|\mathcal{N}_i|$ for neighbors

edge type-specific affine function



Scoring the Structures

After several convolutional layers, node embeddings are aggregated into a graph level representation vector

logistic sigmoid function

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

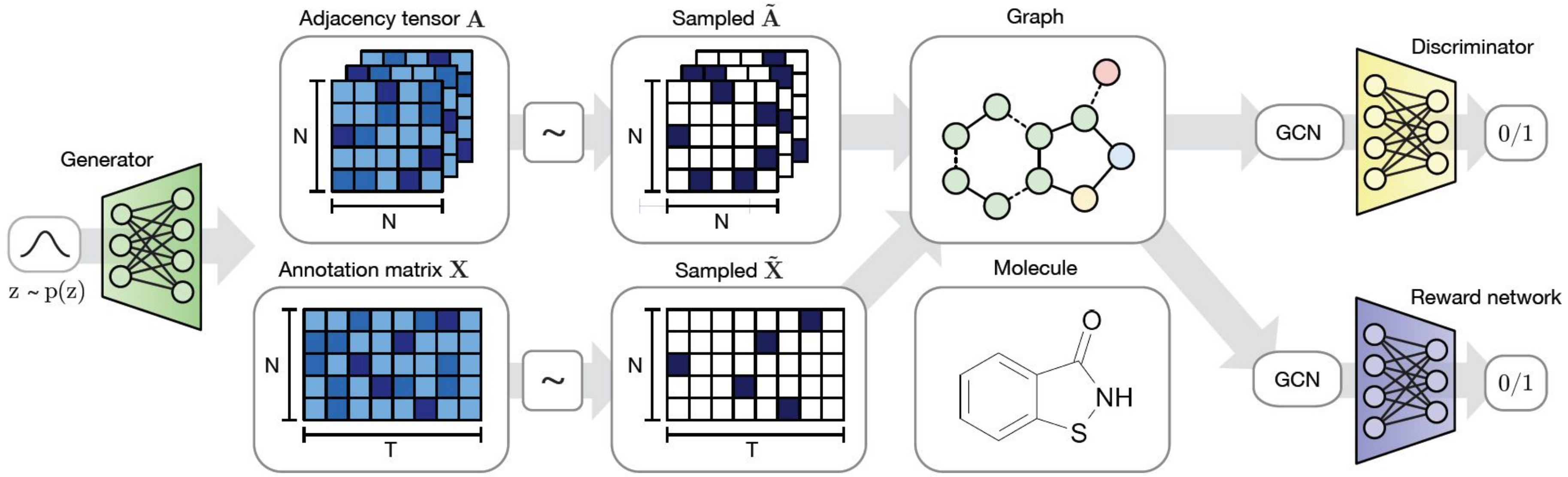
$$\boldsymbol{h}_{\mathcal{G}} = \tanh \boldsymbol{h}'_{\mathcal{G}}$$

MLP (with linear output layer)

Another MLP processes $\boldsymbol{h}_{\mathcal{G}}$ and produces a graph level scalar output

- Discriminator: ∈ (-∞, +∞)
- Reward Network: ∈ (0,1)

Model Full Flow

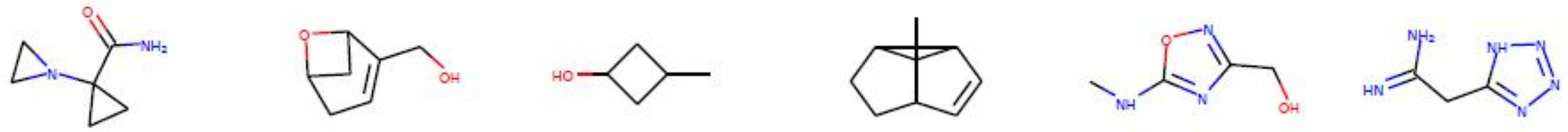

Generator
$z \sim p(z)$
Adjacency tensor $\mathbf{A}$
N
N
Annotation matrix $\mathbf{X}$
N
T
~
~
Sampled $\tilde{\mathbf{A}}$
N
N
Sampled $\tilde{\mathbf{X}}$
N
T
Graph
Molecule
O
NH
S
GCN
Discriminator
0/1
GCN
Reward network
0/1

Experimental Results

Dataset

QM9 Dataset of 133,885 organic compounds composed entirely of carbon (C), hydrogen (H), oxygen (O), nitrogen (N), and fluorine (F) atoms

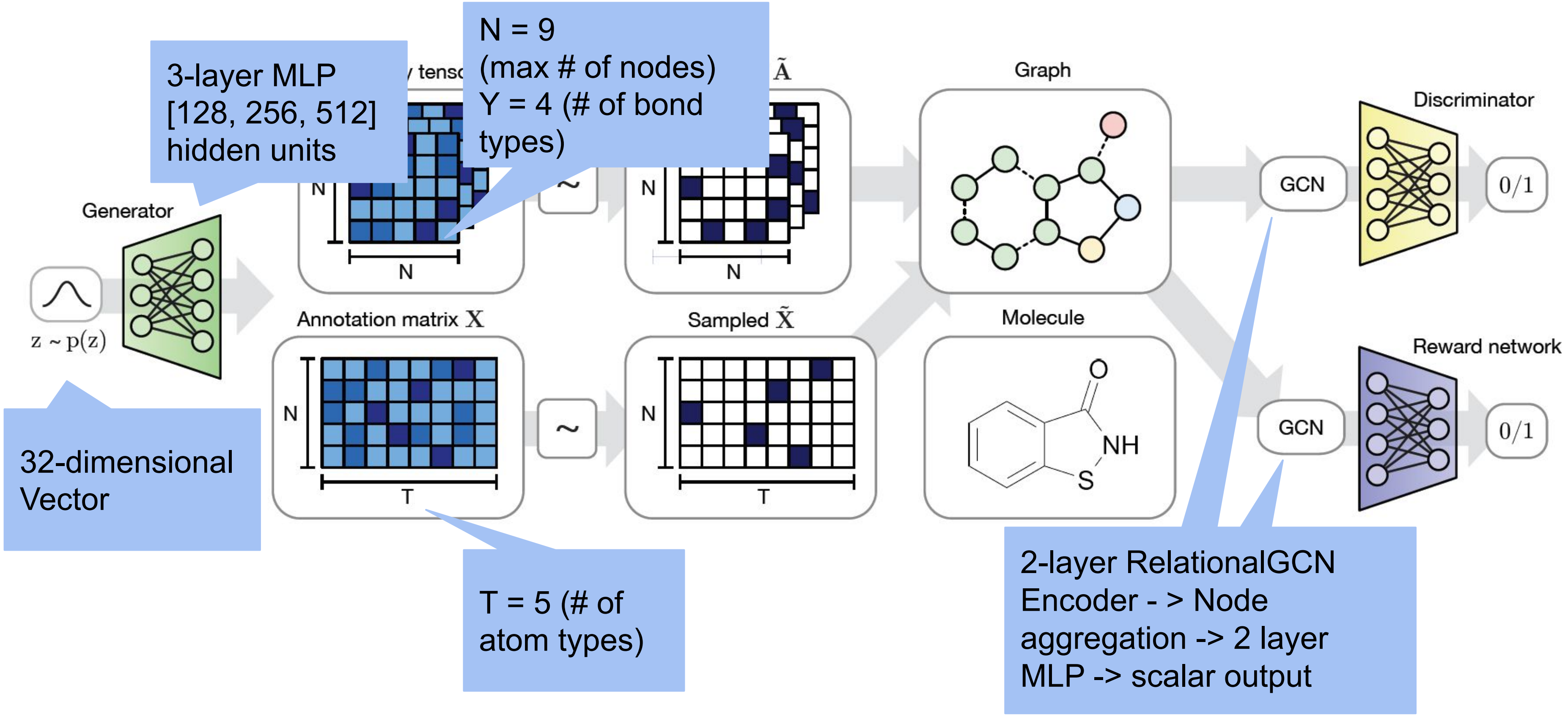
- Each compound has at most 9 heavy atoms (CONF)



Method Evaluation Metrics

- Validity
  - Valid structures / All generated structures
- Novelty
  - Valid structures not in training dataset / All valid generated structures
- Uniqueness
  - Unique valid structures / Total valid structures

Model Architecture



3-layer MLP [128, 256, 512] hidden units

N = 9 (max # of nodes)
Y = 4 (# of bond types)

32-dimensional Vector

T = 5 (# of atom types)

2-layer RelationalGCN Encoder - > Node aggregation -> 2 layer MLP -> scalar output

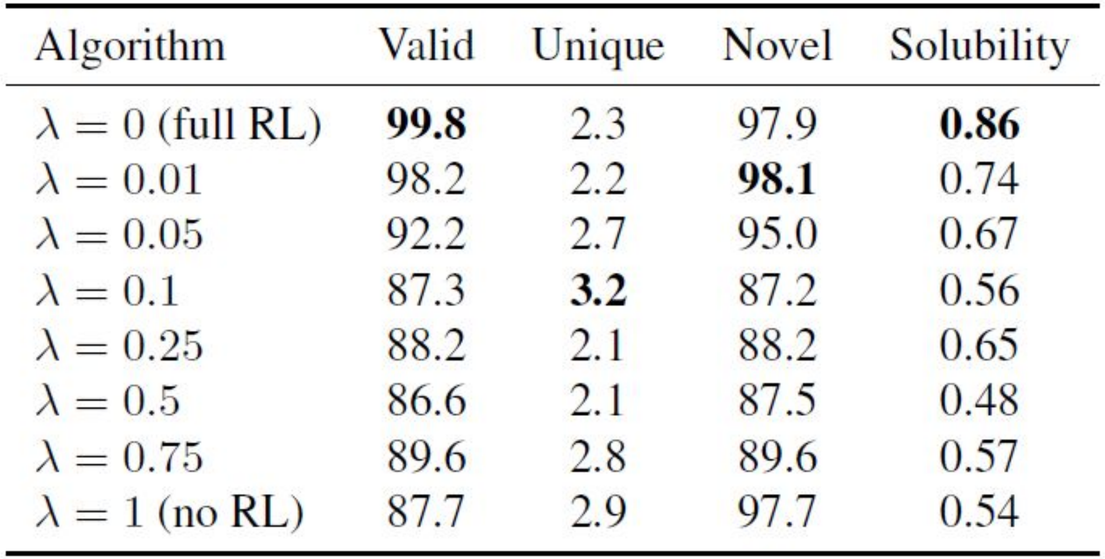
Training

- Batch size: 32
- Adam Optimizer, Learning Rate = 10^{-3}
- Reward network requires several epochs of pre-training
  - Generator trained for first half of training with only WGAN objective
  - Combined loss used in second half of training

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

- Employ early stopping to avoid complete mode collapse

Experiment 1: λ Hyperparameter Optimization

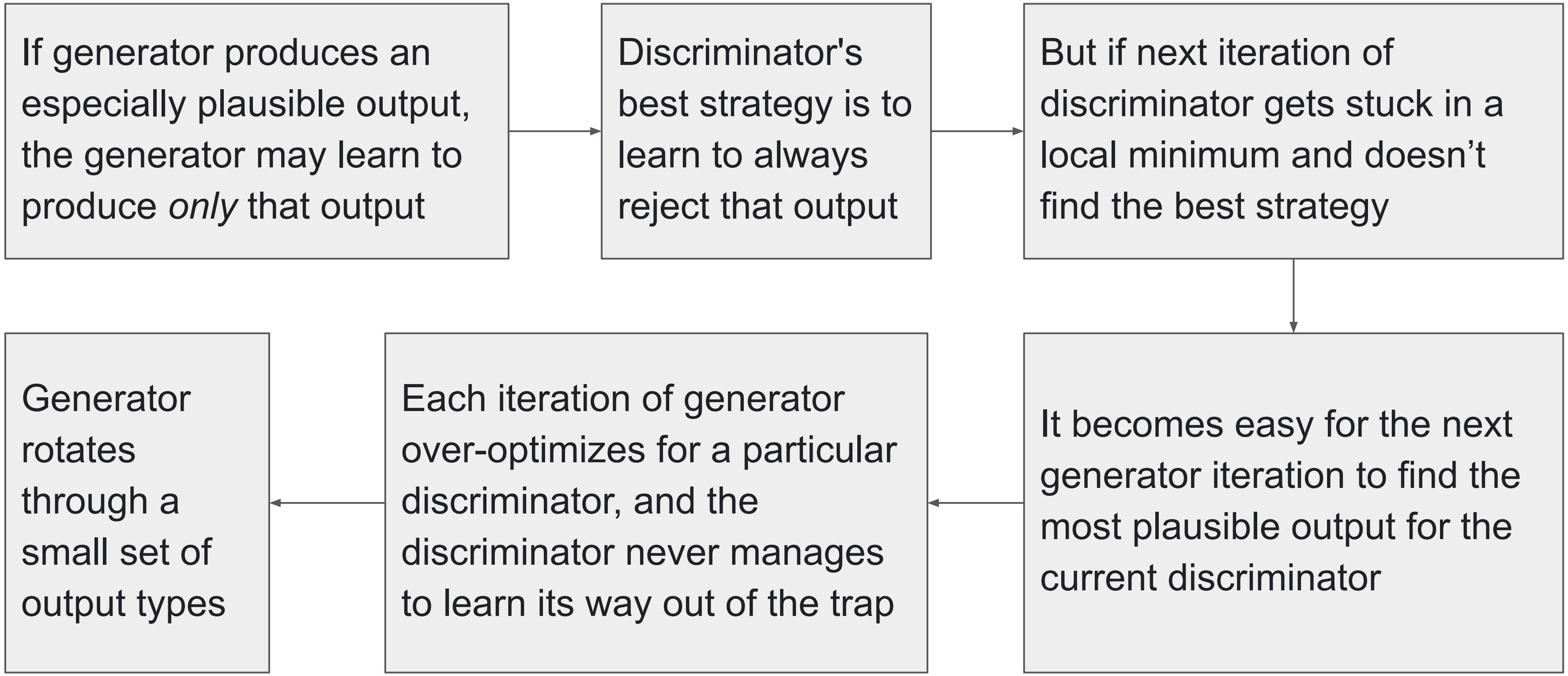


Algorithm	Valid	Unique	Novel	Solubility
$\lambda = 0$ (full RL)	**99.8**	2.3	97.9	**0.86**
$\lambda = 0.01$	98.2	2.2	**98.1**	0.74
$\lambda = 0.05$	92.2	2.7	95.0	0.67
$\lambda = 0.1$	87.3	**3.2**	87.2	0.56
$\lambda = 0.25$	88.2	2.1	88.2	0.65
$\lambda = 0.5$	86.6	2.1	87.5	0.48
$\lambda = 0.75$	89.6	2.8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

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

Result: only reward loss is necessary

Mode Collapse in GANs



If generator produces an especially plausible output, the generator may learn to produce *only* that output

Discriminator's best strategy is to learn to always reject that output

But if next iteration of discriminator gets stuck in a local minimum and doesn’t find the best strategy

It becomes easy for the next generator iteration to find the most plausible output for the current discriminator

Each iteration of generator over-optimizes for a particular discriminator, and the discriminator never manages to learn its way out of the trap

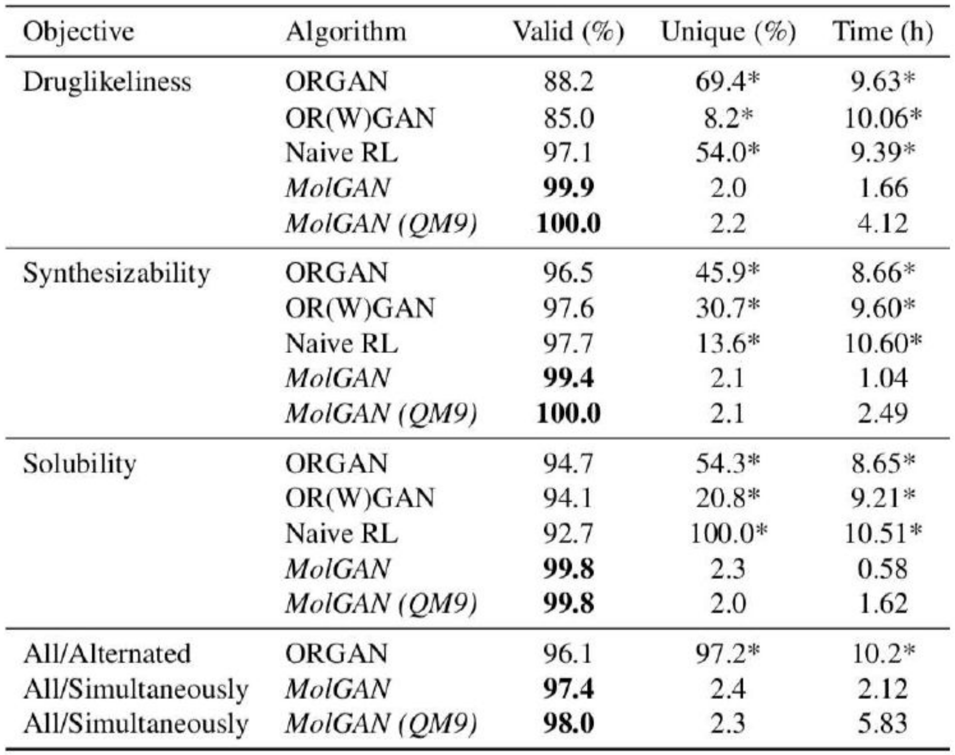
Generator rotates through a small set of output types

https://developers.google.com/machine-learning/gan/problems

Generated Molecule Evaluation Metrics

- Druglikeness
  - How likely a compound is to be a drug
- Synthesizability
  - How easy is a molecule to synthesize
- Solubility
  - The degree to which a molecule is hydrophilic

Experiment 2: Comparison to ORGAN

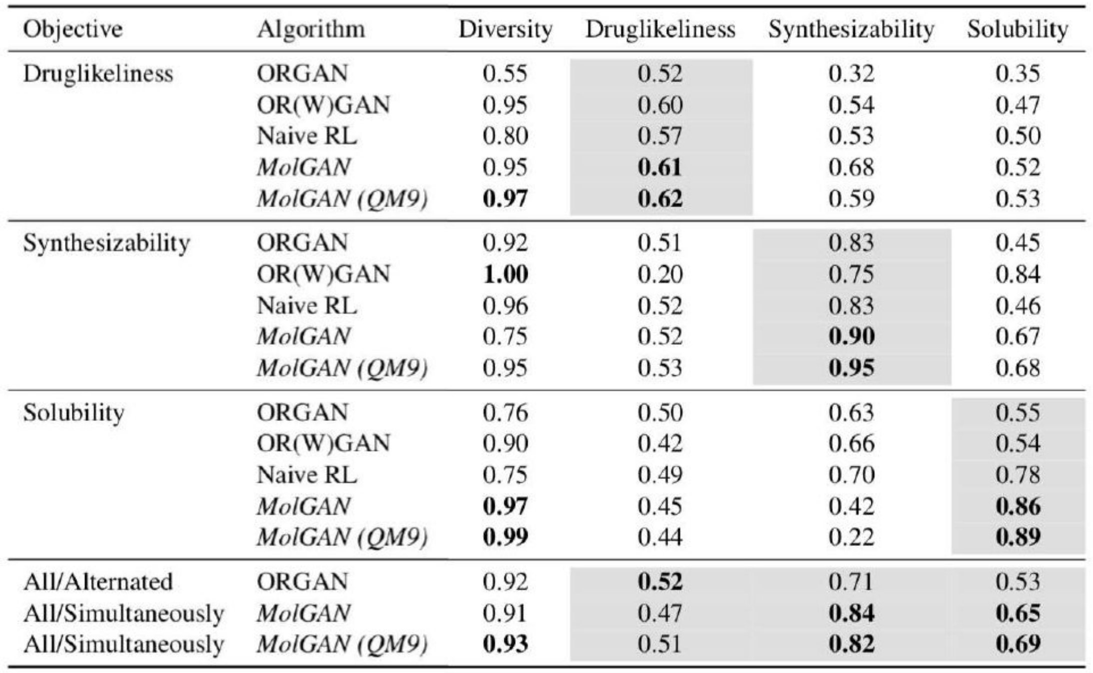


Objective	Algorithm	Valid (%)	Unique (%)	Time (h)
Druglikeliness	ORGAN	88.2	69.4*	9.63*
	OR(W)GAN	85.0	8.2*	10.06*
	Naive RL	97.1	54.0*	9.39*
	MolGAN	**99.9**	2.0	1.66
	MolGAN (QM9)	**100.0**	2.2	4.12
Synthesizability	ORGAN	96.5	45.9*	8.66*
	OR(W)GAN	97.6	30.7*	9.60*
	Naive RL	97.7	13.6*	10.60*
	MolGAN	**99.4**	2.1	1.04
	MolGAN (QM9)	**100.0**	2.1	2.49
Solubility	ORGAN	94.7	54.3*	8.65*
	OR(W)GAN	94.1	20.8*	9.21*
	Naive RL	92.7	100.0*	10.51*
	MolGAN	**99.8**	2.3	0.58
	MolGAN (QM9)	**99.8**	2.0	1.62
All/Alternated	ORGAN	96.1	97.2*	10.2*
All/Simultaneously	*MolGAN*	**97.4**	2.4	2.12
All/Simultaneously	*MolGAN (QM9)*	**98.0**	2.3	5.83

- Validity
  - Others: 85 - 97%
  - **MolGAN: 98 - 100%**
- Uniqueness
  - Others: 10 - 70%
  - **MolGAN: 2%**
- Time consumption
  - **□ - ½ time required by others**

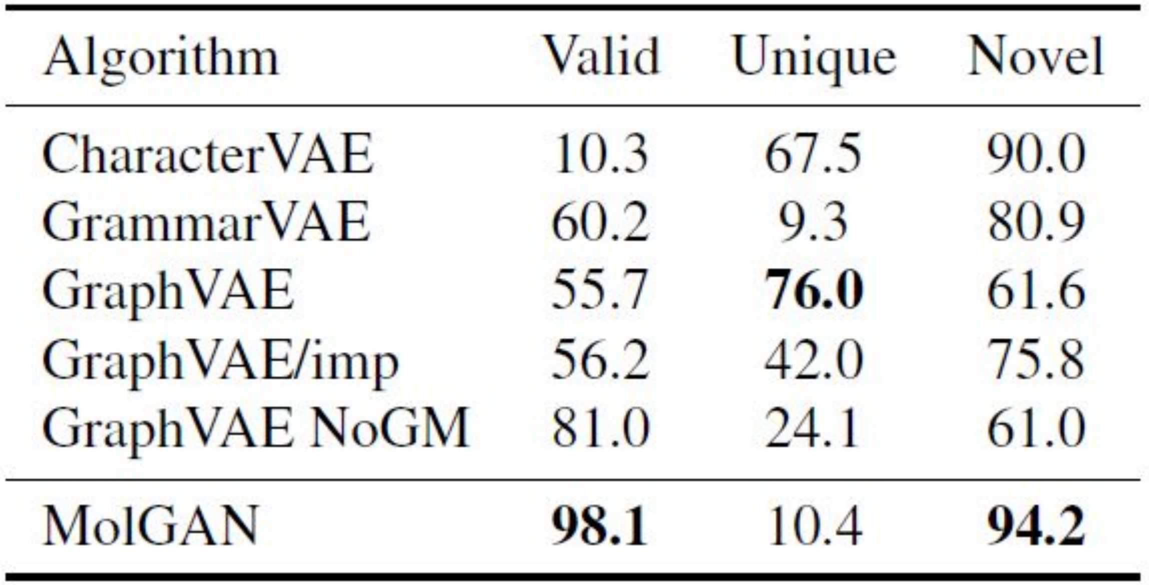
Experiment 2: Comparison to ORGAN

Higher score than other methods for all the properties



Objective	Algorithm	Diversity	Druglikeliness	Synthesizability	Solubility
Druglikeliness	ORGAN	0.55	0.52	0.32	0.35
	OR(W)GAN	0.95	0.60	0.54	0.47
	Naive RL	0.80	0.57	0.53	0.50
	MolGAN	0.95	**0.61**	0.68	0.52
	MolGAN (QM9)	**0.97**	**0.62**	0.59	0.53
Synthesizability	ORGAN	0.92	0.51	0.83	0.45
	OR(W)GAN	**1.00**	0.20	0.75	0.84
	Naive RL	0.96	0.52	0.83	0.46
	MolGAN	0.75	0.52	**0.90**	0.67
	MolGAN (QM9)	0.95	0.53	**0.95**	0.68
Solubility	ORGAN	0.76	0.50	0.63	0.55
	OR(W)GAN	0.90	0.42	0.66	0.54
	Naive RL	0.75	0.49	0.70	0.78
	MolGAN	**0.97**	0.45	0.42	**0.86**
	MolGAN (QM9)	**0.99**	0.44	0.22	**0.89**
All/Alternated	ORGAN	0.92	**0.52**	0.71	0.53
All/Simultaneously	*MolGAN*	0.91	0.47	**0.84**	**0.65**
All/Simultaneously	*MolGAN (QM9)*	**0.93**	0.51	**0.82**	**0.69**

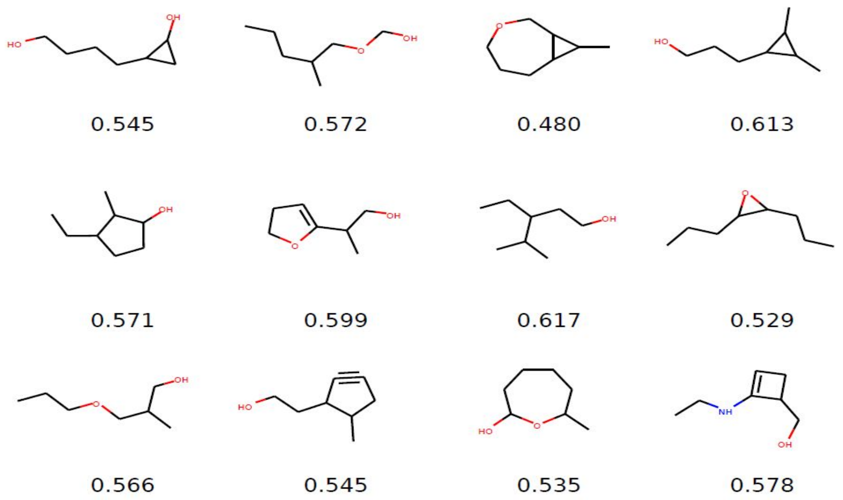
Experiment 3: Comparison to other methods

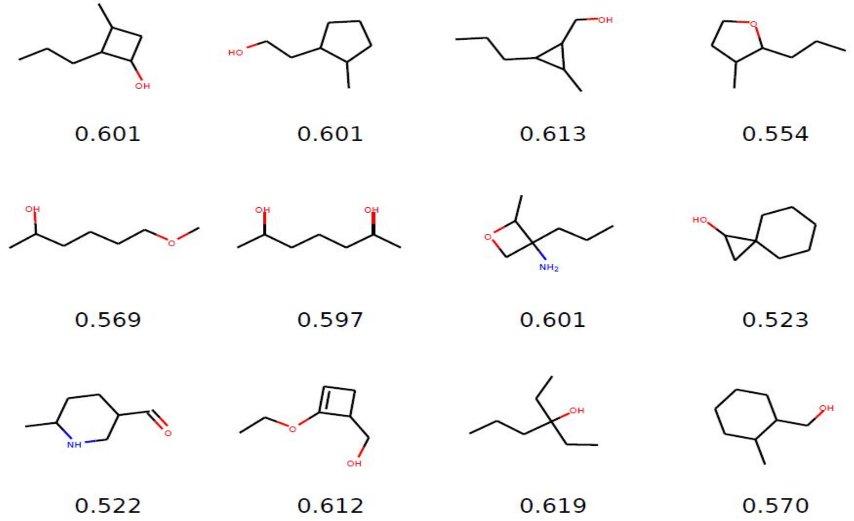


Algorithm	Valid	Unique	Novel
CharacterVAE	10.3	67.5	90.0
GrammarVAE	60.2	9.3	80.9
GraphVAE	55.7	**76.0**	61.6
GraphVAE/imp	56.2	42.0	75.8
GraphVAE NoGM	81.0	24.1	61.0
MolGAN	**98.1**	10.4	**94.2**

MolGAN achieves both high validity and novelty scores

Examples of Generated Molecules





※numbers: druglikeness (QED score)

Discussion

Pros:

- Very high (~100%) valid output structure ratio
- GraphNN + RL is effective for biochemical optimization
- Light computational cost, fast learning

Cons:

- Mode collapse - same structure is repeatedly generated
  - Normalization techniques (e.g. spectral normalization) could help
- Fixed atom count

Demo

Demo Outputs

```
 100% (1040/1040) [################################################################################] ETA: 0:00:00
2021-02-17 16:16:57 Epochs         60/100 in 1:50:52 (last epoch in 0:01:05), ETA: 1:13:54
2021-02-17 16:18:27 Validation --> {'NP score': 0.9581788729387261,
 'QED score': 0.474503668468447,
 'SA score': 0.2689629926191282,
 'diversity score': 0.6573819264126546,
 'drugcandidate score': 0.40128474446261014,
 'la': 1.0,
 'logP score': 0.3217449740331785,
 'loss D': -117.52902,
 'loss G': 65.90267,
 'loss RL': -0.7108812,
 'loss V': 0.67149824,
 'novel score': 57.13231677324152,
 'unique score': 21.815051647811117,
 'valid score': 81.31999969482422}
2021-02-17 16:18:49 Model saved in trained_model!
```