Interpretable Latent Variable Model for Molecular Design

UGP Presentation

Gurpreet Singh

25th April, 2019

Indian Institute of Technology, Kanpur

Background

BBVI - Overview

In Variational Inference, the optimization objective is given by the ELBO

$$\log p\left(\mathbf{x}\right) \geq \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{\mathbf{z} \sim q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \Big[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \,\big|\, \mathbf{x}, \boldsymbol{\phi}) \Big]$$

BBVI - Overview

In Variational Inference, the optimization objective is given by the ELBO

$$\log p\left(\mathbf{x}\right) \geq \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \Big[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}) \Big]$$

Black Box Variational Inference is based on stochastic optimization of the variational objective

$$\nabla \mathcal{L}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\phi}} \log q(\mathbf{z} \mid \mathbf{x}, \phi) \Big(\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\phi}) \Big) \Big]$$

The above term is known as the Score Function.

BBVI - Reparameterization

If we can find a reparametrization $\boldsymbol{\epsilon} = f^{-1}(\mathbf{z}, \boldsymbol{\phi})$ such that $\boldsymbol{\epsilon}$ is independent of $\boldsymbol{\phi}$, then

$$\nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\phi}) = \mathop{\mathbb{E}}_{q(\boldsymbol{\epsilon})} \left[\nabla_{\mathbf{z}} \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}) \right]_{\mathbf{z}=f(\boldsymbol{\epsilon}, \boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}} f(\boldsymbol{\epsilon}, \boldsymbol{\phi}) \right]$$

BBVI - Reparameterization

If we can find a reparametrization $\boldsymbol{\epsilon} = f^{-1}(\mathbf{z}, \boldsymbol{\phi})$ such that $\boldsymbol{\epsilon}$ is independent of $\boldsymbol{\phi}$, then

$$\nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\phi}) = \mathop{\mathbb{E}}_{q(\boldsymbol{\epsilon})} \left[\nabla_{\mathbf{z}} \left[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\phi}) \right]_{\mathbf{z}=f(\boldsymbol{\epsilon}, \boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}} f(\boldsymbol{\epsilon}, \boldsymbol{\phi}) \right]$$

If $\mathbf{z}$ is discrete $p(\mathbf{z})$ is a p.m.f. and therefore, the derivatives no longer exist. Hence, reparameterization is **not** possible

Discrete Latent Variables

Some methods to train discrete latent variable models -

1. Marginalize all discrete latent variables
2. Using local expectation gradients, and reparametrization and marginalization
3. Relaxed computation of discrete densities (straight through approach)
4. Smooth relaxations for discrete variables (e.g. Concrete Distribution)
5. Using the Score Function approach with variance reduction using control variates/non-uniform sampling.

The Concrete Distribution

Sampling from a categorical distribution: Sample U from a $[0, 1]$ uniform distribution, then reparameterize as

$$\mathrm{C} = \underset{k \in [K]}{\arg\max} \ \log \alpha_k - \log\left(-\log\left(\mathrm{U}_k\right)\right)$$

A softmax version of this can be used as smooth relaxations

$$\mathrm{Z}_k = \frac{\exp\left(\left(\log \alpha_k + G_k\right)/\tau\right)}{\sum_{k'=1}^{K} \exp\left(\left(\log \alpha_{k'} + G_{k'}\right)/\tau\right)}$$

where τ is the **temperature** parameter.

The Gumbolt Trick

RBM Prior

Gumbel-Softmax trick only works for factorial priors. The Gumbolt trick offer an approach which allows us to work with more expressive priors, the (Restricted) Boltzmann Machine priors.

Boltzmann Machine Prior

$$q\left(\mathbf{z}\right) = \frac{e^{-E(\mathbf{z})}}{Z_\theta}, \qquad \text{where}$$
$$E(\mathbf{z}) = \mathbf{z}^{\mathrm{T}}\mathbf{b} + \mathbf{z}^{\mathrm{T}}\mathrm{W}\mathbf{z}$$

Restricted Boltzmann Machine Prior

$$E(\mathbf{z}) = \mathbf{z}_1{}^{\mathrm{T}}\mathbf{b}_1 + \mathbf{z}_2{}^{\mathrm{T}}\mathbf{b}_2 + \mathbf{z}_1{}^{\mathrm{T}}\mathrm{W}\mathbf{z}_2$$

Gumbolt - Relaxed ELBO

The posterior and the prior for the smoothing variables are given as

$$p\left(\boldsymbol{\zeta} \,|\, \boldsymbol{\theta}\right) = \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})}}{\widetilde{Z}_{\boldsymbol{\theta}}}, \text{ where } \widetilde{Z}_{\boldsymbol{\theta}} = \int_{\boldsymbol{\zeta}} e^{E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})} \, \mathrm{d}\boldsymbol{\zeta}$$

$$q\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) = \prod_i \zeta_i q_i + (1 - \zeta_i)(1 - q_i)$$

Gumbolt - Relaxed ELBO

The posterior and the prior for the smoothing variables are given as

$$p\left(\boldsymbol{\zeta} \,|\, \boldsymbol{\theta}\right) = \frac{e^{-E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})}}{\widetilde{Z}_{\boldsymbol{\theta}}}, \text{ where } \widetilde{Z}_{\boldsymbol{\theta}} = \int_{\boldsymbol{\zeta}} e^{E_{\boldsymbol{\theta}}(\boldsymbol{\zeta})} \, \mathrm{d}\boldsymbol{\zeta}$$

$$q\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) = \prod_i \zeta_i q_i + (1 - \zeta_i)(1 - q_i)$$

The alternate ELBO, therefore, is given as

to $$\widetilde{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \mathop{\mathbb{E}}_{q(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi})} \left[\log p\left(\mathbf{x} \,|\, \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log q\left(\boldsymbol{\zeta} \,|\, \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log \widetilde{Z}_{\boldsymbol{\theta}}$$

Gumbolt - Relaxed ELBO

Relaxed ELBO is given as

to

$$\breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{q(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \left[\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log q\left(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log Z_{\boldsymbol{\theta}}$$

Gumbolt - Relaxed ELBO

Relaxed ELBO is given as

$$\breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{q(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \left[\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log q\left(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log Z_{\boldsymbol{\theta}}$$

to

We have

$$\log p\left(\mathbf{x} \mid \boldsymbol{\theta}\right) \geq \widetilde{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) \geq \breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right)$$

Gumbolt - Relaxed ELBO

Relaxed ELBO is given as

$$\breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) = \underset{q(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi})}{\mathbb{E}} \left[\log p\left(\mathbf{x} \mid \boldsymbol{\zeta}, \boldsymbol{\theta}\right) - \log q\left(\boldsymbol{\zeta} \mid \mathbf{x}, \boldsymbol{\phi}\right) - E_{\boldsymbol{\theta}}(\boldsymbol{\zeta}) \right] - \log Z_{\boldsymbol{\theta}}$$

to

We have

$$\log p\left(\mathbf{x} \mid \boldsymbol{\theta}\right) \geq \widetilde{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right) \geq \breve{\mathcal{L}}\left(\mathbf{x}, \boldsymbol{\phi}\right)$$

Also, as $\tau \to 0$, $\breve{\mathcal{L}} \to \widetilde{\mathcal{L}}$.
Since as $\tau \to 0$, we have $\boldsymbol{\zeta} \to \mathbf{z}$, therefore $\breve{\mathcal{L}} \to \widetilde{\mathcal{L}} \to \mathcal{L}$

Gumbolt - Regeneration

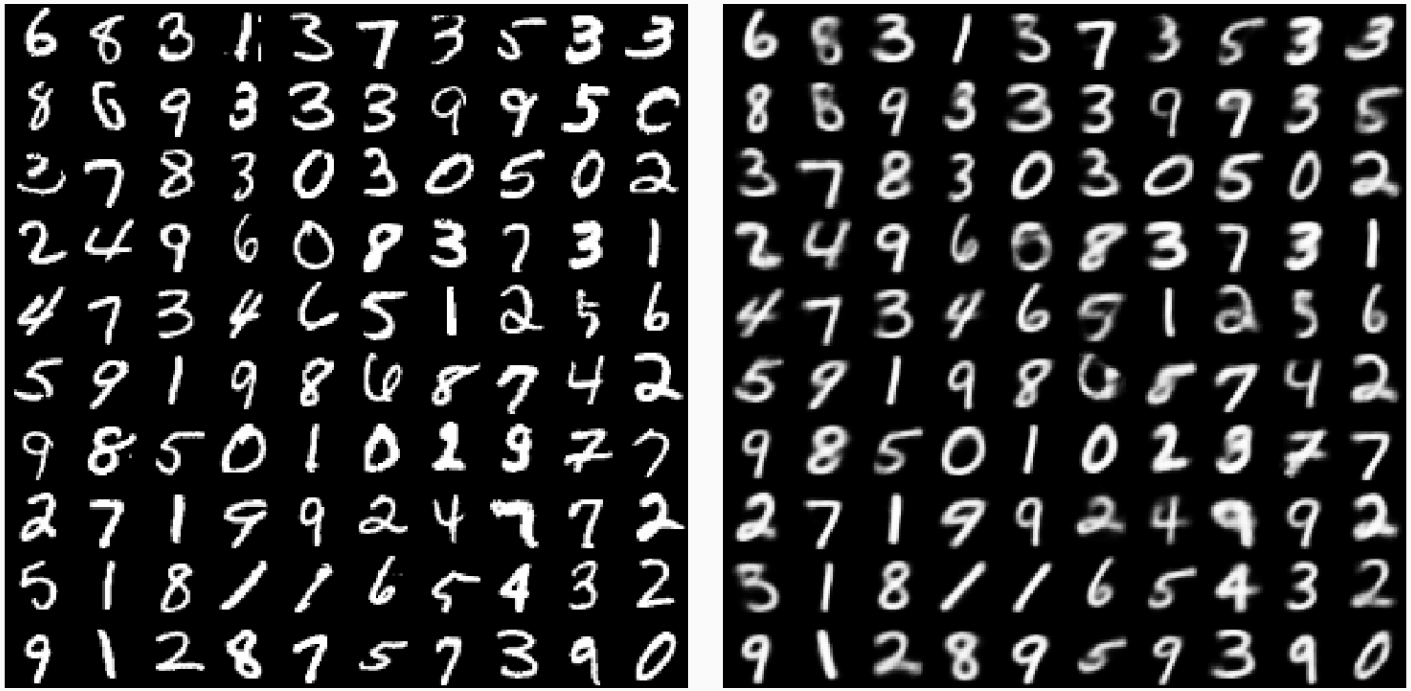


Figure 1: Regerenation Plot for Gumbolt-Vae

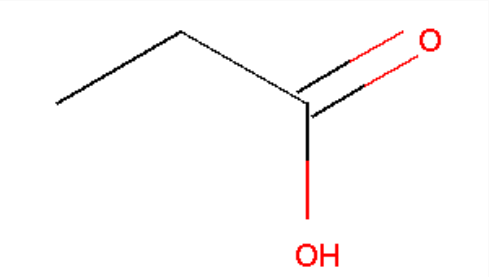
Gumbolt - Sampling

Figure 2: Sampling Plot for Gumbolt-Vae

Molecular Design

Molecular Representations

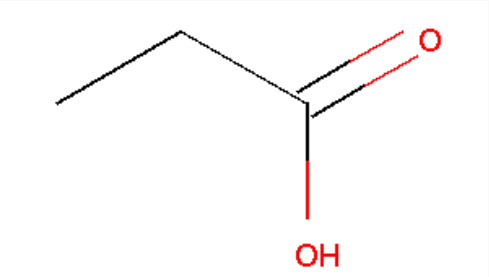
1. SMILES - offers a string representation of the molecule's skeletal structure



The SMILES string of the above molecule is given as "CCC(=O)O"

Molecular Representations

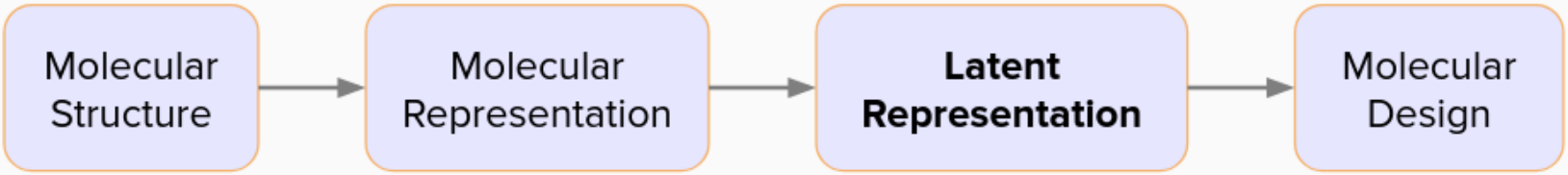
1. SMILES - offers a string representation of the molecule's skeletal structure



   The SMILES string of the above molecule is given as "CCC(=O)O"

2. Molecular Graphs - assumes each atom to be a node and each bond to be an edge.

Automatic Molecular Design



Many deep learning based generative models have been proposed lately for automatic molecule design. Two common approaches are GANs and VAEs (our focus). Two categorizations -

1. SMILES based models
2. Graph based model

SMILES based models

1. **CVAE** – Modifies vanilla VAEs to use convolutional layers in the encoder and layers of GRU networks for the decoder to predict sequences of strings (SMILES)

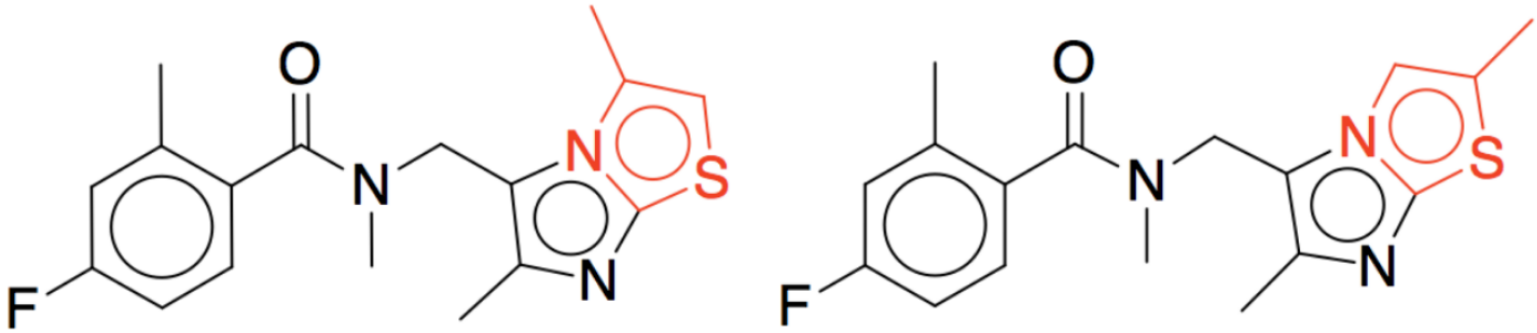
SMILES based models

1. **CVAE** – Modifies vanilla VAEs to use convolutional layers in the encoder and layers of GRU networks for the decoder to predict sequences of strings (SMILES)
2. **GVAE** – Improves upon CVAE by utilizing the parse trees (CFG) generated from SMILES strings as the molecular representaiton and using Grammar Variational Autoencoders as their architecture. Generates syntactically correct SMILES strings.

SMILES based models

1. **CVAE** – Modifies vanilla VAEs to use convolutional layers in the encoder and layers of GRU networks for the decoder to predict sequences of strings (SMILES)
2. **GVAE** – Improves upon CVAE by utilizing the parse trees (CFG) generated from SMILES strings as the molecular representaiton and using Grammar Variational Autoencoders as their architecture. Generates syntactically correct SMILES strings.
3. **SD-VAE** – Incorporates semantics into the parse trees (Attribute Grammars) to generate syntactically as well as semmantically coherent SMILES strings.

Problem with SMILES



Cc1cn2c(CN(C)C(=O)c3ccc(F)cc3C)c(C)nc2s1

Cc1cc(F)ccc1C(=O)N(C)Cc1c(C)nc2scc(C)n12

Molecular Graph based models

1. **VAE with Regularization Framework** – Uses Variational Graph Autoencoders with additional constraints on graph connectivity and edge types

Molecular Graph based models

1. **VAE with Regularization Framework** – Uses Variational Graph Autoencoders with additional constraints on graph connectivity and edge types
2. **CGVAE** – Generates molecular graphs using sequential graph generation utilizing Gated Graph Neural Networks in both the encoder and the decoder.

Molecular Graph based models

1. **VAE with Regularization Framework** – Uses Variational Graph Autoencoders with additional constraints on graph connectivity and edge types
2. **CGVAE** – Generates molecular graphs using sequential graph generation utilizing Gated Graph Neural Networks in both the encoder and the decoder.

Problem: Invalid intermediate structures while sequential generation

Junction Tree Variational Autoencoder (JT-VAE)

JT-VAE generates molecular graphs in two phases

1. It generates a tree-structured object (a junction tree) that represents the scaffold of subgraph components and their coarse relative arrangements

Junction Tree Variational Autoencoder (JT-VAE)

JT-VAE generates molecular graphs in two phases

1. It generates a tree-structured object (a junction tree) that represents the scaffold of subgraph components and their coarse relative arrangements
2. The subgraphs (nodes in the junction tree) are assembled into a coherent molecule graph using a graph message passing network. This approach incrementally generates the molecular graph while maintaining molecular validity at every step.

Junction Tree Variational Autoencoder (JT-VAE)

JT-VAE generates molecular graphs in two phases

1. It generates a tree-structured object (a junction tree) that represents the scaffold of subgraph components and their coarse relative arrangements
2. The subgraphs (nodes in the junction tree) are assembled into a coherent molecule graph using a graph message passing network. This approach incrementally generates the molecular graph while maintaining molecular validity at every step.

Thus, the JT-VAE encoder has two parts: graph encoder and tree encoder, so has the decoder: tree decoder and graph decoder. The graph and tree encoders are closely related to message passing neural networks (MPNNs)

JT-VAE (Continued)

- The molecular structure is encoded into a two-part continuous latent representation $\mathbf{z} = (\mathbf{z}_{\mathrm{T}}, \mathbf{z}_{\mathrm{G}})$ where $\mathbf{z}_{\mathrm{T}}$ encodes the tree structure and what the subgraph components are in the tree. $\mathbf{z}_{\mathrm{G}}$ encodes the graph to capture the fine-grained connectivity

JT-VAE (Continued)

- The molecular structure is encoded into a two-part continuous latent representation $\mathbf{z} = (\mathbf{z}_{\mathrm{T}}, \mathbf{z}_{\mathrm{G}})$ where $\mathbf{z}_{\mathrm{T}}$ encodes the tree structure and what the subgraph components are in the tree. $\mathbf{z}_{\mathrm{G}}$ encodes the graph to capture the fine-grained connectivity
- Both the molecular graph and junction tree are encoded via message passing using adapted gated recurrent units (GRUs).

JT-VAE (Continued)

- The molecular structure is encoded into a two-part continuous latent representation $\mathbf{z} = (\mathbf{z}_{\mathrm{T}}, \mathbf{z}_{\mathrm{G}})$ where $\mathbf{z}_{\mathrm{T}}$ encodes the tree structure and what the subgraph components are in the tree. $\mathbf{z}_{\mathrm{G}}$ encodes the graph to capture the fine-grained connectivity
- Both the molecular graph and junction tree are encoded via message passing using adapted gated recurrent units (GRUs).
- The junction tree is sequentially generated with message passing in a top-down as well as bottom-up (Contains topological and label predictions)

JT-VAE (Continued)

- The molecular structure is encoded into a two-part continuous latent representation $\mathbf{z} = (\mathbf{z}_T, \mathbf{z}_G)$ where $\mathbf{z}_T$ encodes the tree structure and what the subgraph components are in the tree. $\mathbf{z}_G$ encodes the graph to capture the fine-grained connectivity
- Both the molecular graph and junction tree are encoded via message passing using adapted gated recurrent units (GRUs).
- The junction tree is sequentially generated with message passing in a top-down as well as bottom-up (Contains topological and label predictions)
- Graph decoding is performed in an alternating manner where a subgraph (for each node of the tree) is updated using messages generated by the tree encode

Gumbolt JT-VAE

Gumbolt Junction Tree VAE

Binary latent vectors ($\mathbf{b}$) are used in company with real latent vectors ($\mathbf{r}$), and the Gumbolt trick is employed to reparameterize the binary latent vectors (with RBM prior). Final latent representation $\mathbf{z}$ is given as $\mathbf{z} = \mathbf{b} \odot \mathbf{r}$

Gumbolt JT-VAE: Architecture

The Decoder

$$[\,\mathbf{b}_1, \mathbf{b}_2\,] \sim \mathrm{RBM}\left(\mathbf{a}_1, \mathbf{a}_2, \mathrm{W}\right), \qquad \mathbf{r} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

The Encoder

$$q\left(\mathbf{b}_n, \mathbf{r}_n \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \prod_{k=1}^{K} q\left(b_{n,k} \,\middle|\, \mathbf{x}_n, \mathrm{A}, \boldsymbol{\phi}\right) q\left(r_{n,k} \,\middle|\, \mathbf{x}_n, \mathbf{b}_n, \mathrm{A}, \boldsymbol{\phi}\right)$$

and

$$q\left(b_{n,k} \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \mathrm{Bernoulli}\left(\pi_{n,k}\right)$$
$$q\left(r_{n,k} \,\middle|\, \mathbf{x}, \mathrm{A}, \boldsymbol{\phi}\right) = \mathcal{N}\left(\mu_{n,k}, \sigma_{n,k}^2\right)$$

where $\{\pi_{n,k}\}_{k=1}^{K} = \mathrm{FC}\left(\mathbf{h}_n\right)$, $\{\mu_{n,k}, \sigma_{n,k}\}_{k=1}^{K} = \mathrm{FC}\left(\mathbf{h}_n, \mathbf{b}_n\right)$ and $\mathbf{h}_n = \mathrm{MPNN}\left(\mathbf{x}_n, \mathrm{A}\right)$

Comparison with JT-VAE (ZINC Dataset)

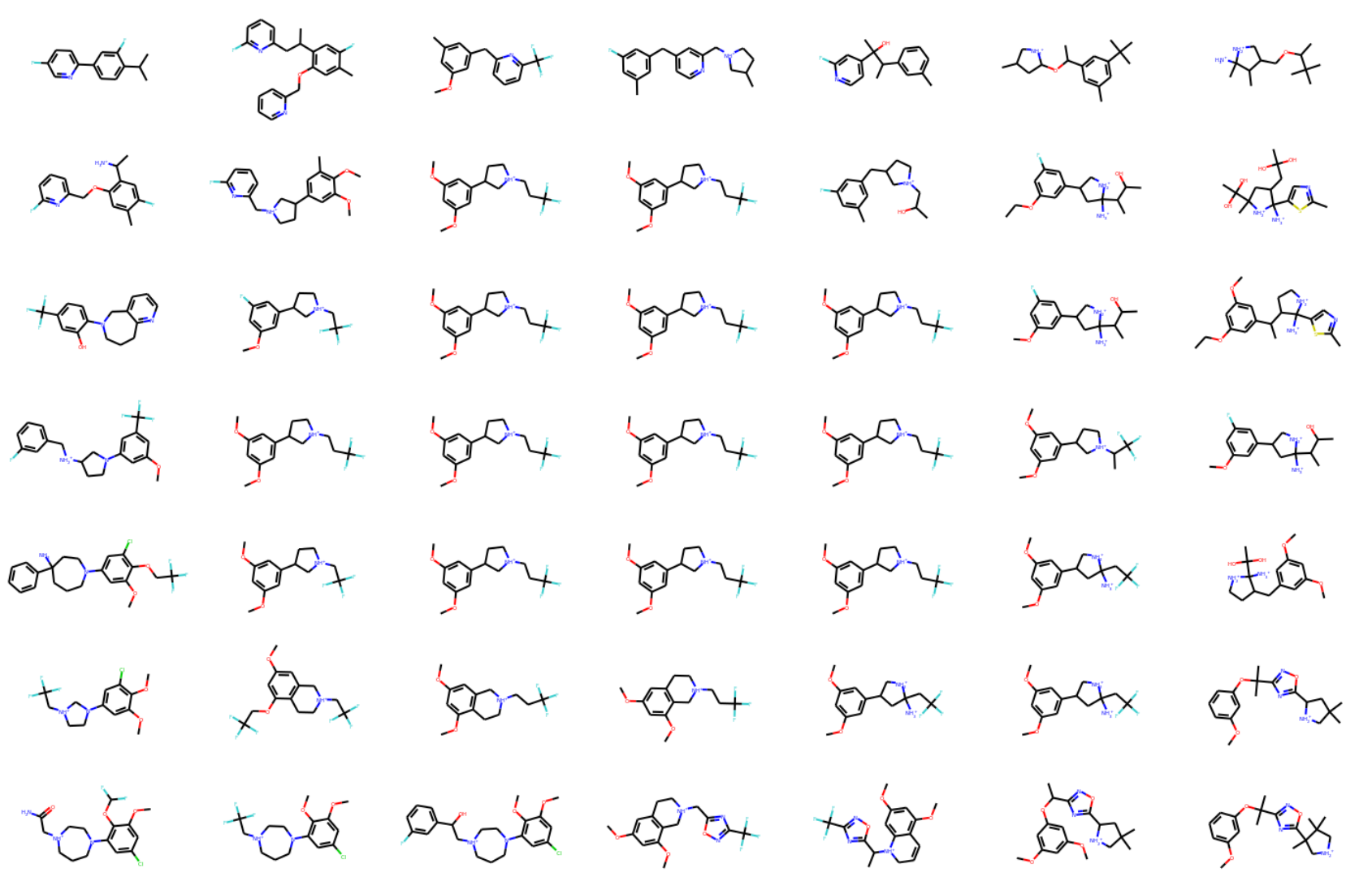
Gumbolt JT-VAE	JT-VAE
3.889	4.019

Table 1: Reconstruction Loss

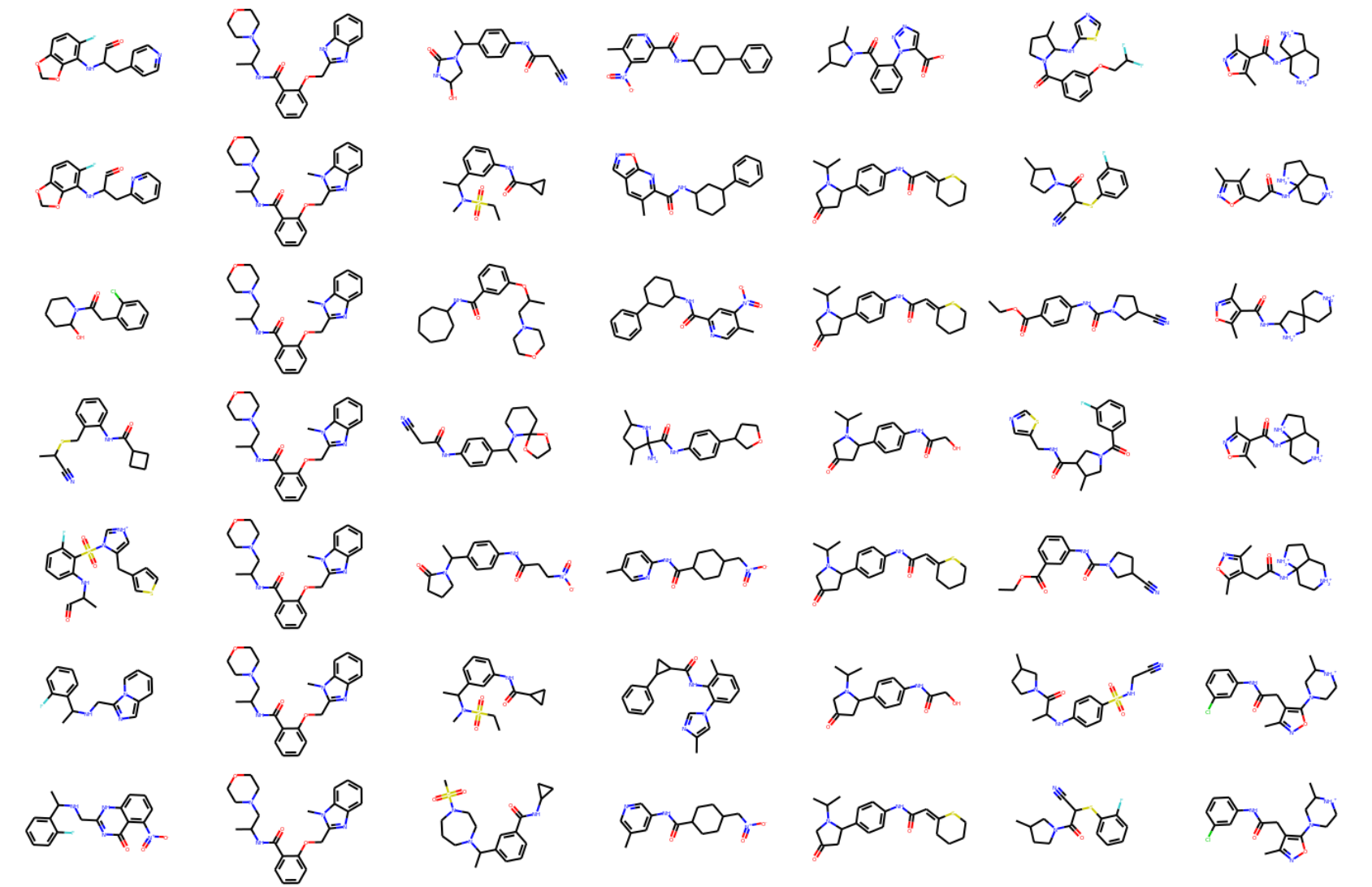
	Gumbolt JT-VAE	JT-VAE
Topological Accuracy	93.38	**94.16**
Classification Accuracy	**99.21**	97.57
Assembly Accuracy	**95.34**	94.31

Table 2: Model Accuracies

Qualitative Analysis - Neighborhood Sampling



Qualitative Analysis - Sampling from the Prior



Further Possible Extensions

1. Replace Teacher Forcing with Scheduled Sampling
2. Avoid choosing a node as root in the Junction Tree