Our Team

DOW TEAM:
Ellen Du, Joey Storer
Sukrit Mukhopadhyay
Matthew Christianson
Hein Koelman, Ryan Marson
William Edsall, Bart Rijksen
Jonathan Moore
Christopher Roth, Peter Margl
Clark Cummins, Dave Magley

NVIDIA TEAM/Alumni:
Abe Stern, Michelle Gill,
John Ashley, Alex Volkov
Outline

Problem statement

Efforts in generative molecular deep learning methods

Our approach
  • Hardware/software
  • Tooling
  • Data curation
  • Model Training and convergence
  • Latent space analysis and inference
  • Generative capability evaluation
Can a molecular generative deep learning system be trained to deliver new molecular designs relevant to our research needs?
Introduction: Generative Molecular Systems

Challenges:
• Molecular encoding (Canonical SMILES)
• Molecular descriptors (100’s)
• Vastness of chemical search space \(10^{60}\)
• Unknown structure/property relationships \(f(n)\)
• Promise of the latent space dimensionality (32-bit)
• Limits on data set used for training (ChEMBL, ZINC)
• Organization of target properties within the latent space (AlogP)
• Molecule discovery workflow (post-filtering)
Attraction of Molecular VAE/GANs

Convert discrete molecules to continuous latent representations
• Molecules are discrete entities
• Subtle molecular transformations have large differences in performance

Undocumented benefit to using negative data in ml/dl
• Availability of a molecular structure axis in DL that is not generally available to ML
• Tendency in science to “move on” relative to negative or poor results

General intro on methods: VAEs

Generally there are numerous methods appearing in the open literature:

• Chemical VAE
• Grammar VAE
• Junction Tree
• ATNC RL
• FC-NN (NVIDIA-Dow)

The best way to go is not entirely clear.

Junction Tree – may be best because of the more natural graph representation – but it may constrain diversity

FC-NN is potentially more efficient.
### Inferencing Comparison to Literature

<table>
<thead>
<tr>
<th>Method</th>
<th>Reconstruction</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Knowns</td>
<td>Inferenced(unknown)</td>
</tr>
<tr>
<td>Chem-VAE</td>
<td>44 %</td>
<td>1 % lit.</td>
</tr>
<tr>
<td>Dow-Chem-VAE</td>
<td>94 %</td>
<td>10 %</td>
</tr>
<tr>
<td>Grammar-VAE</td>
<td>54 %</td>
<td>7 % lit.</td>
</tr>
<tr>
<td>SD-VAE</td>
<td>76 %</td>
<td>44 % lit.</td>
</tr>
<tr>
<td>Graph-VAE</td>
<td>-</td>
<td>14 % lit.</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>77 %</td>
<td>100 % lit.</td>
</tr>
<tr>
<td>Dow-FC-NN</td>
<td>90 %</td>
<td>-- %</td>
</tr>
<tr>
<td>ATNC-RL</td>
<td>-</td>
<td>71 % lit.</td>
</tr>
</tbody>
</table>
Models and Training Details
Model Details: Architectures Explored

3 Variational AutoEncoders

- chemVAE

- Junction Tree VAE
  Jin et al, 2018 (MIT)

- Fully convolutional VAE
  (NVIDIA-Dow)

Similar in setup
Different in details
## Model Details: Differences In Inputs

<table>
<thead>
<tr>
<th></th>
<th>chemVAE</th>
<th>fcVAE</th>
<th>jtVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Smiles</td>
<td>Smiles</td>
<td>Molecular Graph</td>
</tr>
<tr>
<td></td>
<td>ClC1c[nH]cn1</td>
<td>ClC1c[nH]cn1</td>
<td><img src="image" alt="Molecular Graph" /></td>
</tr>
</tbody>
</table>

**Graph:**
- ClC1c[nH]cn1

**SMILES:**
- ClC1c[nH]cn1

**One-hot encoding:**

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>C</th>
<th>c</th>
<th>1</th>
<th>c</th>
<th>nH</th>
<th>c</th>
<th>n</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cl</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**JT.nn Encoder output**
- $h_{T_G}$
- $h_{G}$

**Molecular vector**
- $Z_G$

**New Molecular vector**
- $Z_G$

**New Tree vector**
- $Z_T$

$z = [Z_T, Z_G]$
## Model Details: Differences In Sequence Modeling

<table>
<thead>
<tr>
<th></th>
<th>chemVAE</th>
<th>fcnVAE</th>
<th>jtnn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer type used</td>
<td>Teachers forcing</td>
<td>Residual block</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>for sequence</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td>modeling</td>
<td><em>Lamb et al, 2016</em></td>
<td><em>Bai et al, 2018</em></td>
<td><em>Cho et al, 2014</em></td>
</tr>
</tbody>
</table>
Model Training Details: Data Compilation

Types of Hetero-atoms

CHEMBL Database (1.4×10^6)

Screening using Designed Metrics

Molecules Similar to Dow Set (118,000)
Model Training Details: Hardware

NVIDIA DGX-1

NVIDIA DGX-1 Specifications

<table>
<thead>
<tr>
<th><strong>NVIDIA DGX-1 Specifications</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPUs</strong></td>
</tr>
<tr>
<td><strong>GPUs</strong></td>
</tr>
<tr>
<td><strong>System Memory</strong></td>
</tr>
<tr>
<td><strong>GPU Memory</strong></td>
</tr>
<tr>
<td><strong>Storage</strong></td>
</tr>
<tr>
<td><strong>Networking</strong></td>
</tr>
<tr>
<td><strong>Power</strong></td>
</tr>
<tr>
<td><strong>Size</strong></td>
</tr>
<tr>
<td><strong>GPU Throughput</strong></td>
</tr>
</tbody>
</table>
Model Training Details: Software Environment

Container: Docker container

Standard
Lightweight
Secure

Packages

Chemistry: RDKit, DeepChem
Data Processing: Numpy, Pandas, Rapids
ML/DL: SciKitLearn, Keras, Tensorflow, Pytorch, XGBoost
Tuning/Scaling Up: Hyperopt, Horovod
Model Training Details: Hyperparameter Optimization

Hyperopt
Model Training Details: Distributed Model Training

- Data Parallelism
- Network Optimal
- User friendly
Model Training Details: Latent Space Organization
Generative Capability Evaluation
Hit Rate Analysis ( > 0 hits/1000 attempts)

ChEMBL TEST = 11800 test molecules inferenced (1000 attempts)

<table>
<thead>
<tr>
<th>Model</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-VAE</td>
<td>94.4 %</td>
</tr>
<tr>
<td>55550</td>
<td></td>
</tr>
<tr>
<td>JT-NN</td>
<td>100 %</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>FC-NN</td>
<td>94 %</td>
</tr>
<tr>
<td>14587*</td>
<td></td>
</tr>
</tbody>
</table>

C-VAE
655 TEST molecules not decoding
VAE Hit rate: Molecules that never decoded

Analysis of molecules from ChEMBL-TEST (655) that did not decode with 1000 attempts:

1. SMILES string length distribution for the non-decoding molecules

2. Inference study increased to 10,000 attempts/molecule
   a. 549/655 still never decoded
   b. 16% successful decoded at least once on 10,000 additional attempts
   c. One molecule decoded an additional 44 times
Distribution of SMILES string lengths

E. Putin, et al.,
*Mol. Pharmaceutics* 2018, 15, 4386-4397
Distance calculation and performance

GPU enabled—distance matrix calculation:
1. Characterizing latent space
2. Support inferencing
   a. Nearest neighbor analysis
   b. Gaussian process support

Rough method comparison:
(30,000 molecules, 900 x 10^6 distances)
Python (Simple, non-vectorized)
5 x 10^5 (DGX-1)
Scipy.spatial.distance.euclidean
10^4 (DGX-1)
Numba/CUDA
1 (DGX-1)
Latent Space Vectors (Kernel Density Est)
C-VAE, JT-NN, FC-NN

Epoch 55500

C-VAE Latent Space Vector

JT-NN Latent Space Vector
How far apart are the molecules in the Latent Space?

ChEMBL (118,000 molecules)
Select 1000 molecules
Calc. Dist. Matrix & plot

Early epoch 2500
Epoch 55500

Mean = 3.2
Std. dev. = 0.38
Max = 4.8
Min = 0.3
Interpolation from the Latent Space

Linear interpolation
• Stepping through training set and linearly interpolating between endpoints chosen from the training set

Spherical-linear interpolation
• Stepping through training set and spherical-linearly interpolating between endpoints chosen from the training set

Hyperspheres
• Utilizing the distance matrix to select point for expanding hyperspheres

Comment on JT-NN BO search & expand approach
Molecular Interpolation in a Continuous Design Space

LERP/SLERP

Algorithm only chose points across the whole of the training set (118,000 molecules) and then interpolated between points in ranges to ensure that, at a minimum, each molecule became an end-point for interpolation.
Inferencing followed by molecular filtering

- Inferenced 15,000,000
  - Unique and Valid SMILES 1,500,000
    - Too many (F, Cl, Br, I) -90 %
    - Too many rings -21 %
    - Too many X-H -1 %
    - Too many O, N -3 %
    - Too many RotBnds -97 %
    - Too Many Other -90 %

300
Synthetic Accessibility Score

The SAScore across:

INPUT: ChEMBL (118,000)
OUTPUT: Inferenced_e55500
TEST: Dow

Conclusions

C-VAE

Chem-VAE modeled after Bombarelli works better than reported and delivers good molecules. The time/epoch is high and the number of epochs needed is \( \sim 50,000 \).

JT-NN

Junction Tree converges faster, is a more natural representation of molecules, and delivers good molecules.

FC-NN

Fully Convolutional works well, converges faster than C-VAE, and delivers good molecules.