Learning a Continuous Representation of 3D Molecular Structures with Deep Generative Models

Matthew Ragoza, Tomohide Masuda, David Ryan Koes

Department of Computational & Systems Biology, University of Pittsburgh, Pittsburgh, PA

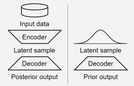
Abstract

Machine learning in drug discovery has been focused on virtual screening of molecular libraries using discriminative models. Generative models are an entirely different approach that learn to represent and optimize molecules in a continuous latent space. These methods have been increasingly successful at generating 2D molecules as SMILES strings and molecular graphs.

In this work, we describe deep generative models of 3D molecular structures using atomic density grids and a novel fitting algorithm for converting continuous grids to discrete molecular structures. Our models jointly represent drug-like molecules and their conformations in a latent space that can be explored through interpolation. We are also able to sample diverse sets of molecules based on a given input compound and increase the probability of creating valid, drug-like molecules.

Background

In a **variational autoencoder** (VAE), an encoder network is trained to map the input data to a set of latent variables that follow a predetermined distribution, then a decoder maps back to the input. This allows outputs to be decoded from either the posterior or prior.



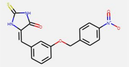
Variational autoencoder posterior and prior sampling

In **generative adversarial networks** (GANs), a discriminative network is trained to tell apart real and fake data while a generative network is concurrently trained to produce outputs that the discriminator mistakenly classifies as real.

Training generative adversarial networks

Deep generative models have been used for generating molecules by representing them as **SMILES strings** and **molecular graphs**. These have mostly been 2D approaches, and have had to overcome the challenges of permutation invariance and invalid outputs.

```
O=C1NC(=S)N/C1=C/
c1cccc(OCc2ccc([N
+](=O)[O-])cc2)c1
```



SMILES string and equivalent molecular graph

Atomic density grids can represent 3D molecular structures and are permutation invariant, though not rotation invariant. A central challenge is converting continuous grids to valid, discrete molecular structures.

3D molecule represented as an atomic density grid

Data sets

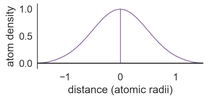
- MolPort training set
  - 7,559,852 molecules
  - ~14 conformers each
  - Hold out random 10% for validation
- PubChem test set
  - 10,014 molecules
  - 1 conformer each
  - Sort into bins from 0.0 to 1.0
    - Max similarity to any train molecule

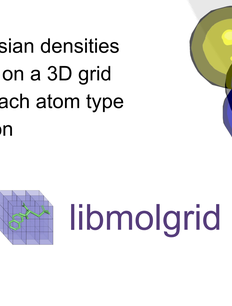
Atom typing

- Assign atom types from
  - Atomic number
  - Aromaticity
  - Hydrophobicity
  - Hydrogen donor/acceptor state
- Using OpenBabel toolkit

Atom gridding

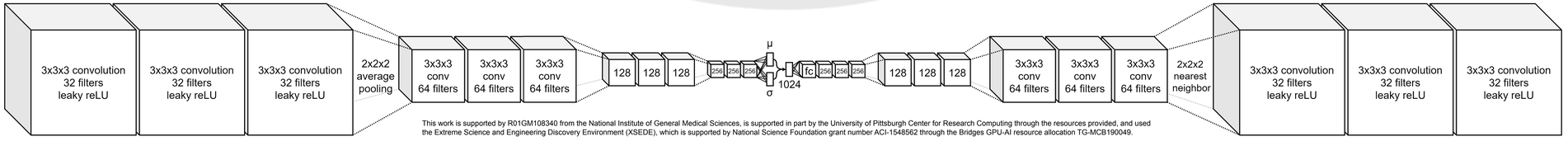
- Represent atoms as Gaussian densities
- Sum density of each atom on a 3D grid
  - Separate channels for each atom type
- libmolgrid GPU acceleration



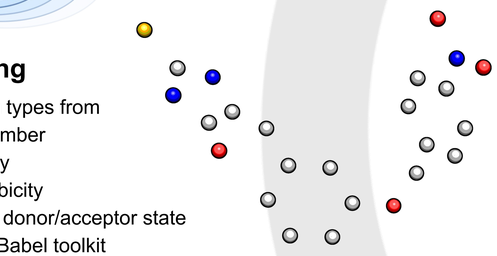


Generative model

VAE + GAN



This work is supported by R01GM108340 from the National Institute of General Medical Sciences, is supported in part by the University of Pittsburgh Center for Research Computing through the resources provided, and used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562 through the Bridges GPU-AI resource allocation TG-MCB190049.

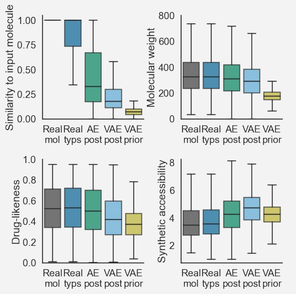


>90%

of generated 3D molecules are valid

Bond adding

- Add bonds based on
  - Atom types
  - Geometry
- Assess validity with RDKit
  - Single fragment
  - No sanitization errors
- Compare properties
  - Similarity to input molecule
  - Molecular weight
  - QED and SA scores



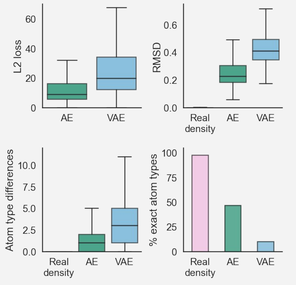
Generated molecules have realistic properties

Atom fitting

- Inverse problem
  - Convert grid to atoms
- Solve with optimization

$$A^* = \arg\min \|\mathbf{G}_{ref} - g(A))\|^2$$

- PyTorch algorithm
  - Atom detection
  - Gradient descent
  - Beam search



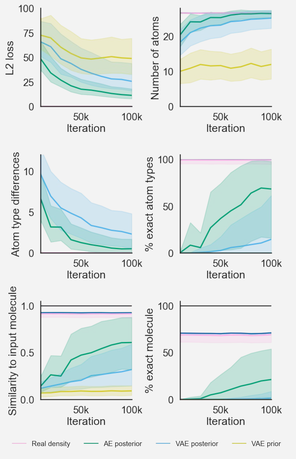
Atom fitting recovers real types and coordinates

Model training

- Train for 100,000 iterations
- Batch size 16
- Adam solver
- Loss function

  = 10.0 * GAN loss
  + 1.0 * L2 loss
  + 0.1 * KL divergence

- Randomly rotate while training
- gnina fork of Caffe framework



Training for L2 loss also improves structures