Descalc Guide

Santiago Vargas

June 2020

1 Introduction

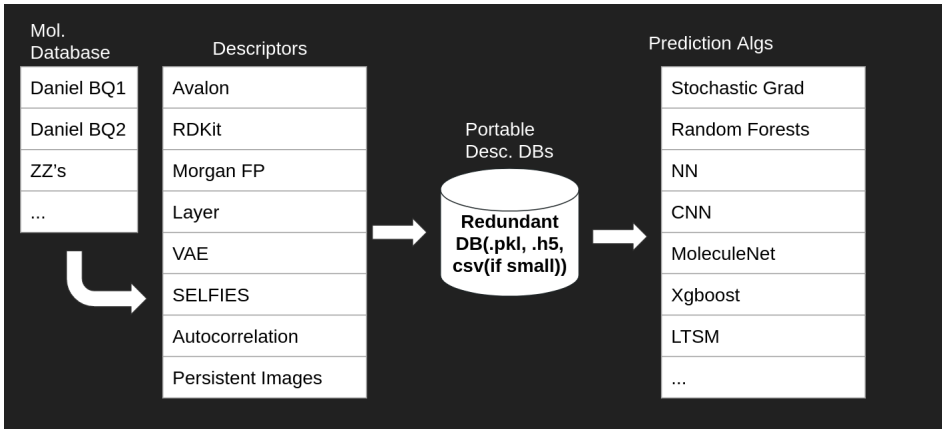

Figure 1: Layout of Descalc and the available descriptors

Descalc is one half of the main program in ML_CO2 package. Descalc pools a set of different descriptor calculators from rdkit, molsimplify, and others to easily allow you to calculate descriptors before training machine learning algorithms on a set of molecules.

1.1 Downloading and Selecting environments

Three environments are included in the package though I was able to get away with two of these alone. Mileage may vary however. I am assuming you have an environment manger such as Anaconda or the one as a part of Pycharm to easily switch between these. The following process is to install a given environment and then switch into it using Anaconda:

```
conda env create -f tf_gpu.yml -n default
conda activate default
python descalc.py --dir [specify dir] --des [specify descriptor]
```

Descriptor	Environment
RDKit	tf_gpu/molsimp
Morgan	tf_gpu/molsimp
Layer	tf_gpu/molsimp
Avalon	tf_gpu/molsimp
SELFIES	tf_gpu /molsimp
VAE	chemvae
Molsimp (Autocorrelation)	molsimp
Persistent Images	molsimp/tf_gpu

Table of environments-descriptor compatibility

2 Descriptor Details

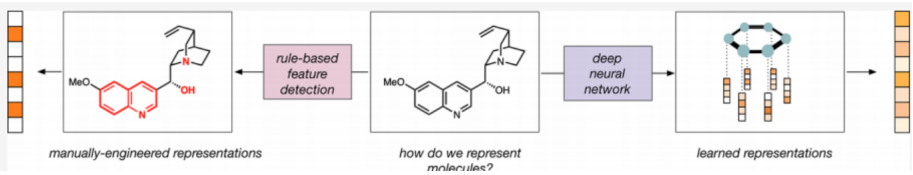

Figure 2: Layout of Descalc and the available descriptors

There are two categories of descriptors currently implemented: those with heuristics encoded and those that let machine learning algorithms optimize description. Keywords for the descriptors are morg, aval, rdkit, layer, persist, vae, self, delta, auto.

2.1 Persistent Images

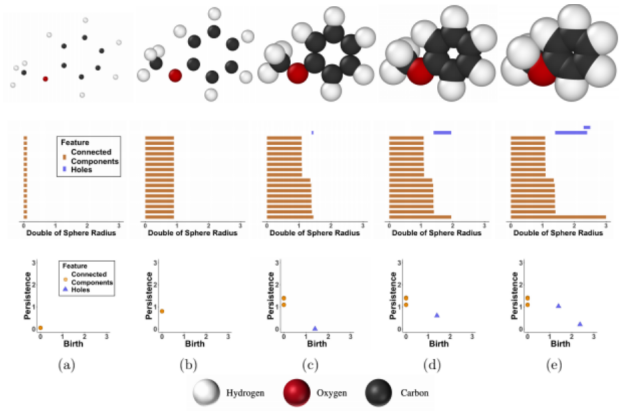

Figure 3: Persistent images use the creation/annihilation of topological features as atoms "grow" to create mappings

Persistent images are a new and relatively untested method. They "grow" atoms from a starting geometry and plot the creation and disappearance of topological features as these atoms grow to create images. Due to the dependence on geometries using cheap methods to generate geometries might not be ideal if coupled with this method.

2.2 Fingerprinting

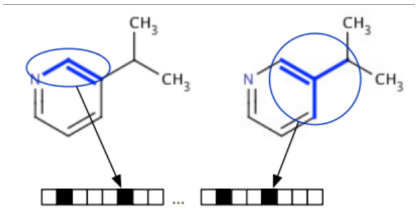

Figure 4: Fingerprinting methods have a set of features they encode for

Fingerprinting methods include rdkit, layer, morgan, and avalon. Each one of these fingerprinting methods is tuned to a different set of features and have been used for a long time. These are a great starting point for any algorithm training due to their intepretability.

2.3 Variational Autoencoder

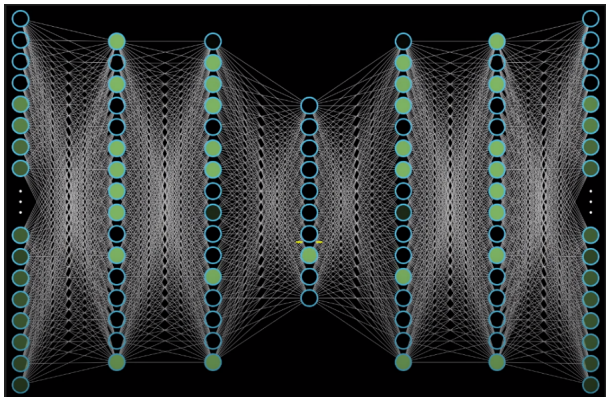

Figure 5: Structure of an Autoencoding Neural Network

Variation autoencoders use the middle layer of a neural network to provide descriptors for a starting geometry. In Fig. 5 this neural network structure has an encoder half and decoder half. This method is useful for optimizing in this latent "descriptor space" and generating new molecules optimized on a given property.

3 Adding to descalc

3.1 New Descriptors

Two main components should be added to added a new descriptor. One would be adding the option to descalc.py:

```
elif (des == "your desc"):
    from <utility file> import <your desc>
    name, mat = <your method>(dir)
```

The other would be adding the corresponding descriptor method into a helper function and importing that utility file.

3.2 New Databases

Descalc can work with sdf, xyz, and smiles files. Note that smiles are converting to 3d structed using openbabels make3d and localopt methods. These use mmff94 (510 steps) forcefields to generate a reasonable guess structure for descriptor calculations. This can (and probably should) be refined to better methods which can be used by descalc in the sdf/xyz format. Depending on what the input to your descriptor is, you may have to convert to xyz, smi,

or sdf file types, there are conversion methods built into the descalc utilities that can deal with these conversions. Simply add your database folder to the corresponding folder in the data directory.

4 Databases

The databases are saved both as .pkl and .h5 formats. As the databases get larger this redundancy is necessary. If there are more than 70000 molecules, descalc separates databases into chunks of 25000 molecules. To load each of these database types the following steps are required.

Pickle:

```
import pandas as pd
...
df_reload = pd.read_pickle("file")
print(df_reload.shape)
```

HDF:

```
import pandas as pd
...
df_reload2 = pd.read_hdf("file")
print(df_reload2.shape)
```

Note, pandas is a database manipulation package that does most of the heavy lifting with larger databases, it's an alternative to using something like numpy. From here you can access the data using the mat and name keyword. There is currently no standardized script for merging descriptor databases to data for fitting but the *xtb_vs_dft* notebook and *process* is a good start at parsing the text files for names.