```
! wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.8.2-Linux-x86_64.sh
! chmod +x Miniconda3-py37_4.8.2-Linux-x86_64.sh
! bash ./Miniconda3-py37_4.8.2-Linux-x86_64.sh -b -f -p /usr/local
! conda install -c rdkit rdkit -y
import sys
sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

```
  setuptools          pkgs/main/linux-64::setuptools-45.2.0-py37_0
  six                 pkgs/main/linux-64::six-1.14.0-py37_0
  sqlite              pkgs/main/linux-64::sqlite-3.31.1-h7b6447c_0
  tk                  pkgs/main/linux-64::tk-8.6.8-hbc83047_0
  tqdm                pkgs/main/noarch::tqdm-4.42.1-py_0
  urllib3             pkgs/main/linux-64::urllib3-1.25.8-py37_0
  wheel               pkgs/main/linux-64::wheel-0.34.2-py37_0
  xz                  pkgs/main/linux-64::xz-5.2.4-h14c3975_4
  yaml                pkgs/main/linux-64::yaml-0.1.7-had09818_2
  zlib                pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3


Preparing transaction: done
Executing transaction: done
installation finished.
WARNING:
    You currently have a PYTHONPATH environment variable set. This may cause
    unexpected behavior when running the Python interpreter in Miniconda3.
    For best results, please verify that your PYTHONPATH only points to
    directories of packages that are compatible with the Python interpreter
    in Miniconda3: /usr/local
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - rdkit


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    blas-1.0                   |              mkl           6 KB
    bzip2-1.0.8                |       h7b6447c_0          78 KB
    ca-certificates-2020.12.8  |       h06a4308_0         121 KB
    cairo-1.14.12              |       h8948797_3         906 KB
    certifi-2020.12.5          |   py37h06a4308_0         141 KB
    conda-4.9.2                |   py37h06a4308_0         2.9 MB
    fontconfig-2.13.0          |       h9420a91_0         227 KB
    freetype-2.10.4            |       h5ab3b9f_0         596 KB
    glib-2.66.1                |       h92f7085_0         2.9 MB
    icu-58.2                   |       he6710b0_3        10.5 MB
    intel-openmp-2020.2        |              254         786 KB
    jpeg-9b                    |       h024ee3a_2         214 KB
    lcms2-2.11                 |       h396b838_0         307 KB
    libboost-1.73.0            |      hf484d3e_11        13.9 MB
    libffi-3.3                 |       he6710b0_2          50 KB
    libpng-1.6.37              |       hbc83047_0         278 KB
    libtiff-4.1.0              |       h2733197_1         449 KB
    libuuid-1.0.3              |       h1bed415_2          15 KB
    libxcb-1.14                |       h7b6447c_0         505 KB
    libxml2-2.9.10             |       hb55368b_3         1.2 MB
    lz4-c-1.9.2                |       heb0550a_3         175 KB
    mkl-2020.2                 |              256        128.3 MB
```

```
mkl-2020.2                 |              256       138.3 MB
mkl-service-2.3.0          |   py37he8ac12f_0          52 KB
mkl_fft-1.2.0              |   py37h23d657b_0         148 KB
```

```
import os
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import Draw, Descriptors
from matplotlib import pyplot as plt
%matplotlib inline
```

```
cd /content/drive/MyDrive/ML_2/project/
```

```
/content/drive/MyDrive/ML_2/project
```

```
df = pd.read_csv('250k_smiles.csv')
from sklearn.model_selection import train_test_split
smiles_train, smiles_test = train_test_split(df["smiles"], random_state=42)
print(smiles_train.shape)
print(smiles_test.shape)
```

```
(187091,)
(62364,)
```

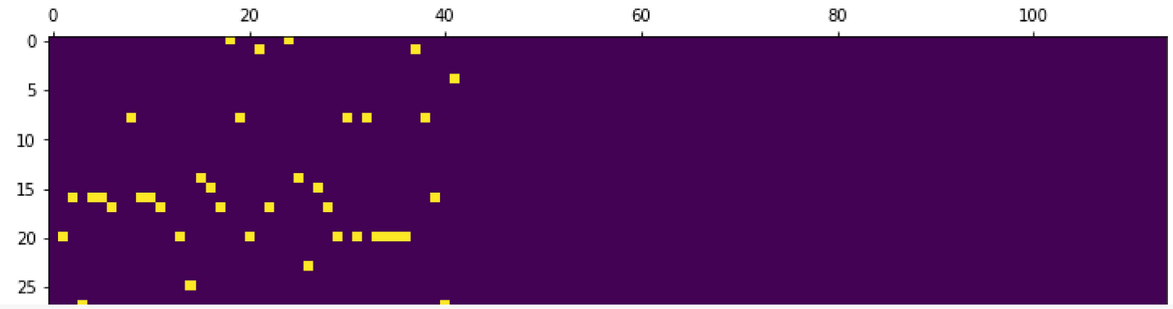
```
charset = set("".join(list(df.smiles))+"!E")
char_to_int = dict((c,i) for i,c in enumerate(charset))
int_to_char = dict((i,c) for i,c in enumerate(charset))
embed = max([len(smile) for smile in df.smiles]) + 5
print (str(charset))
print(len(charset), embed)
```

```
{'N', '2', 'l', '4', '\n', '5', 'r', 'n', ')', '6', 'o', 'S', '\\', 'P', 'H', ']', 'c', '(', 's', '7', 'C', '#', 'O', '+', '8', '@', 'I', '1', '-', 'B', '/', 'F', 'E', '[', '3', '!', '='}
37 115
```

```
def vectorize(smiles):
        one_hot =  np.zeros((smiles.shape[0], embed , len(charset)),dtype=np.int8)
        for i,smile in enumerate(smiles):
            #encode the startchar
            one_hot[i,0,char_to_int["!"]] = 1
            #encode the rest of the chars
            for j,c in enumerate(smile):
                one_hot[i,j+1,char_to_int[c]] = 1
            #Encode endchar
            one_hot[i,len(smile)+1:,char_to_int["E"]] = 1
        #Return two, one for input and the other for output
        return one_hot[:,0:-1,:], one_hot[:,1:,:]
X_train, Y_train = vectorize(smiles_train.values)
X_test,Y_test = vectorize(smiles_test.values)
print (smiles_train.iloc[0])
plt.matshow(X_train[0].T)
#print X_train.shape
```

```
Cc1cc(F)cc([C@H](N)C2([NH+](C)C)CCCC2)c1

<matplotlib.image.AxesImage at 0x7f3457233a20>
```



```
print (X_train.shape)
```

```
(187091, 114, 37)
```

```
"".join([int_to_char[idx] for idx in np.argmax(X_train[0,:,:], axis=1)])
```

```
'!Cc1cc(F)cc([C@H](N)C2([NH+](C)C)CCCC2)c1\nEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE'
```

```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Concatenate
from keras import regularizers
input_shape = X_train.shape[1:]
output_dim = Y_train.shape[-1]
latent_dim = 64
lstm_dim = 64
```

```
unroll = False
encoder_inputs = Input(shape=input_shape)
encoder = LSTM(lstm_dim, return_state=True,
                unroll=unroll)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
states = Concatenate(axis=-1)([state_h, state_c])
neck = Dense(latent_dim, activation="relu")
neck_outputs = neck(states)
```

```
decode_h = Dense(lstm_dim, activation="relu")
decode_c = Dense(lstm_dim, activation="relu")
state_h_decoded =  decode_h(neck_outputs)
state_c_decoded =  decode_c(neck_outputs)
encoder_states = [state_h_decoded, state_c_decoded]
decoder_inputs = Input(shape=input_shape)
decoder_lstm = LSTM(lstm_dim,
                    return_sequences=True,
                    unroll=unroll
                   )
decoder_outputs = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(output_dim, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
#Define the model, that inputs the training vector for two places, and predicts one character ahead of the input
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

```
print (model.summary())
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 114, 37)]    0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 64), (None,  26112       input_2[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 128)          0           lstm_1[0][1]
                                                                 lstm_1[0][2]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           8256        concatenate_1[0][0]
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 114, 37)]    0
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 64)           4160        dense_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 64)           4160        dense_1[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 114, 64)      26112       input_3[0][0]
                                                                 dense_2[0][0]
                                                                 dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 114, 37)      2405        lstm_2[0][0]
==================================================================================================
Total params: 71,205
Trainable params: 71,205
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

```
from keras.callbacks import History, ReduceLROnPlateau
h = History()
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,patience=10, min_lr=0.000001, verbose=1, epsilon=1e-5)
```

```
WARNING:tensorflow:`epsilon` argument is deprecated and will be removed, use `min_delta` instead.
```

```
from keras.optimizers import RMSprop, Adam
opt=Adam(lr=0.005) #Default 0.001
model.compile(optimizer=opt, loss='categorical_crossentropy')

model.fit([X_train,X_train],Y_train,
                    epochs=200,
                    batch_size=256,
                    shuffle=True,
                    callbacks=[h, rlr],
                    validation_data=[X_test,Y_test])
```

```
Epoch 81/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1976 - val_loss: 0.0000e+00

Epoch 00081: ReduceLROnPlateau reducing learning rate to 1.9531249563442543e-05.
Epoch 82/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 83/200
731/731 [==============================] - 15s 20ms/step - loss: 0.1976 - val_loss: 0.0000e+00
Epoch 84/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1973 - val_loss: 0.0000e+00
Epoch 85/200
731/731 [==============================] - 14s 20ms/step - loss: 0.1974 - val_loss: 0.0000e+00
```

```
731/731 [==============================] - 14s 20ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 86/200
731/731 [==============================] - 15s 20ms/step - loss: 0.1973 - val_loss: 0.0000e+00
Epoch 87/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 88/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1975 - val_loss: 0.0000e+00
Epoch 89/200
731/731 [==============================] - 16s 21ms/step - loss: 0.1973 - val_loss: 0.0000e+00
Epoch 90/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 91/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1972 - val_loss: 0.0000e+00

Epoch 00091: ReduceLROnPlateau reducing learning rate to 9.765624781721272e-06.
Epoch 92/200
731/731 [==============================] - 15s 20ms/step - loss: 0.1975 - val_loss: 0.0000e+00
Epoch 93/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 94/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1976 - val_loss: 0.0000e+00
Epoch 95/200
731/731 [==============================] - 15s 21ms/step - loss: 0.1972 - val_loss: 0.0000e+00
Epoch 96/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1972 - val_loss: 0.0000e+00
Epoch 97/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 98/200
731/731 [==============================] - 15s 21ms/step - loss: 0.1972 - val_loss: 0.0000e+00
Epoch 99/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1973 - val_loss: 0.0000e+00
Epoch 100/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 101/200
731/731 [==============================] - 15s 20ms/step - loss: 0.1972 - val_loss: 0.0000e+00

Epoch 00101: ReduceLROnPlateau reducing learning rate to 4.882812390860636e-06.
Epoch 102/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 103/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 104/200
731/731 [==============================] - 15s 20ms/step - loss: 0.1974 - val_loss: 0.0000e+00
Epoch 105/200
731/731 [==============================] - 14s 19ms/step - loss: 0.1970 - val_loss: 0.0000e+00
Epoch 106/200
731/731 [==============================] - 14s 20ms/step - loss: 0.1972 - val_loss: 0.0000e+00
Epoch 107/200
731/731 [==============================] - 15s 21ms/step - loss: 0.1970 - val_loss: 0.0000e+00
```

```
model.save('Lstm.h5')
```

```
plt.plot(h.history["loss"], label="Loss")
plt.plot(h.history["val_loss"], label="Val_Loss")
plt.yscale("log")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f333e683a20>
```

Encoder :

```
smiles_to_latent_model = Model(encoder_inputs, neck_outputs)

smiles_to_latent_model.save("smi2lat.h5")
```

Decoder :

```
latent_input = Input(shape=(latent_dim,))
state_h_decoded_2 =  decode_h(latent_input)
state_c_decoded_2 =  decode_c(latent_input)
latent_to_states_model = Model(latent_input, [state_h_decoded_2, state_c_decoded_2])
latent_to_states_model.save("lat2state.h5")
```

LSTM layer:

```
inf_decoder_inputs = Input(batch_shape=(1, 1, input_shape[1]))
inf_decoder_lstm = LSTM(lstm_dim,
                    return_sequences=True,
                    unroll=unroll,
                    stateful=True
                   )
inf_decoder_outputs = inf_decoder_lstm(inf_decoder_inputs)
inf_decoder_dense = Dense(output_dim, activation='softmax')
inf_decoder_outputs = inf_decoder_dense(inf_decoder_outputs)
sample_model = Model(inf_decoder_inputs, inf_decoder_outputs)
```

```
for i in range(1,3):
    sample_model.layers[i].set_weights(model.layers[i+6].get_weights())
sample_model.save("Mol_model.h5")
sample_model.summary()
```

```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(1, 1, 37)]              0
_________________________________________________________________
lstm_3 (LSTM)                (1, 1, 64)                26112
_________________________________________________________________
dense_5 (Dense)              (1, 1, 37)                2405
=================================================================
Total params: 28,517
Trainable params: 28,517
```

```
Non-trainable params: 0
```

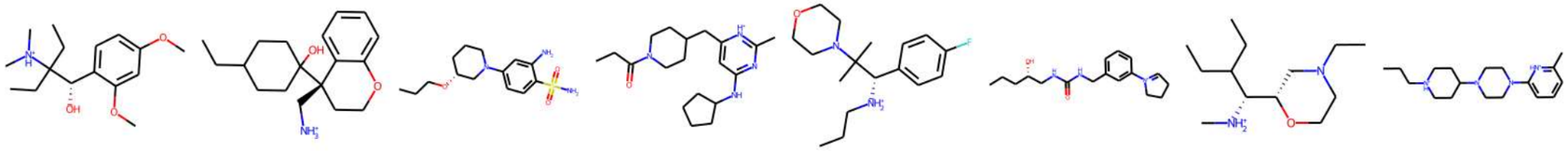
Latest space

```
x_latent = smiles_to_latent_model.predict(X_test)
```

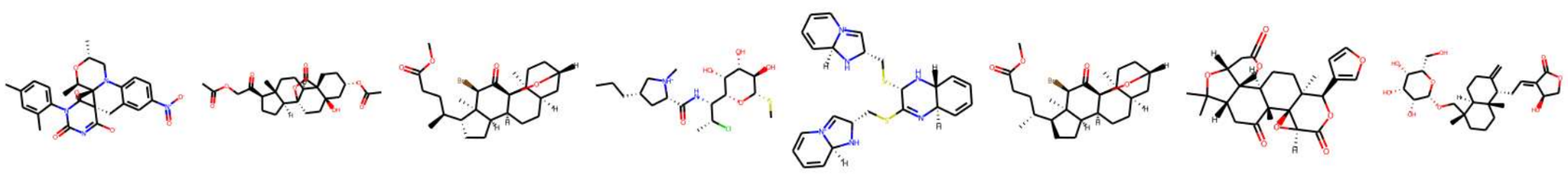
Creating molecules from latent space

```
molno = 5
latent_mol = smiles_to_latent_model.predict(X_test[molno:molno+1])
sorti = np.argsort(np.sum(np.abs(x_latent - latent_mol), axis=1))
print(sorti[0:10])
print(smiles_test.iloc[sorti[0:8]])
Draw.MolsToImage(smiles_test.iloc[sorti[0:8]].apply(Chem.MolFromSmiles))
```

```
[    5 55207 59917 28317 33822 50798 20098 59383 48052 15992]
19753       CCC(CC)([C@@H](O)c1ccc(OC)cc1OC)[NH+](C)C\n
2103       CCC1CCC(O)([C@@]2(C[NH3+])CCOc3ccccc32)CC1\n
10929      CCCO[C@@H]1CCCN(c2ccc(S(N)(=O)=O)c(N)c2)C1\n
73320     CCC(=O)N1CCC(Cc2cc(NC3CCCC3)nc(C)[nH+]2)CC1\n
67465     CCC[NH2+][C@@H](c1ccc(F)cc1)C(C)(C)N1CCOCC1\n
193133      CCC[C@H](O)CNC(=O)NCc1cccc([N+]2=CCCC2)c1\n
67732         CCC(CC)[C@@H]([NH2+]C)[C@@H]1CN(CC)CCO1\n
60031      CCC[NH+]1CCC(N2CCN(c3cccc(C)[nH+]3)CC2)CC1\n
Name: smiles, dtype: object
```



```
Draw.MolsToImage(smiles_test.iloc[sorti[-8:]].apply(Chem.MolFromSmiles))
```

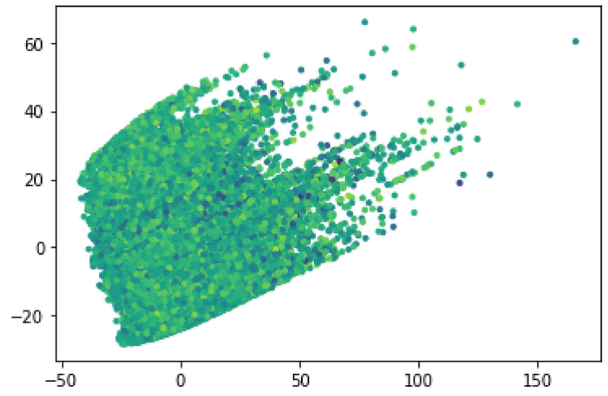


Plotting molecules in latent space :

```
logp = smiles_test.apply(Chem.MolFromSmiles).apply(Descriptors.MolLogP)
from sklearn.decomposition import PCA
```

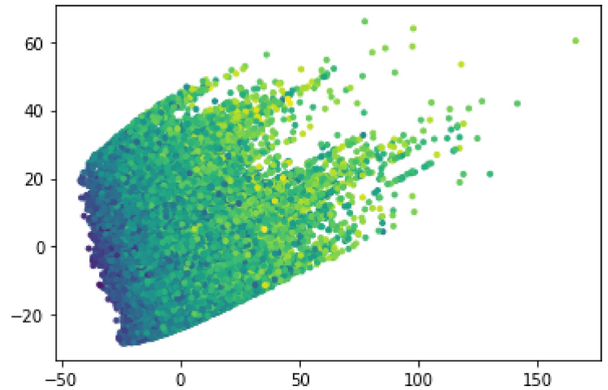
```
pca = PCA(n_components = 2)
red = pca.fit_transform(x_latent)
plt.figure()
plt.scatter(red[:,0], red[:,1],marker='.', c= logp)
print(pca.explained_variance_ratio_, np.sum(pca.explained_variance_ratio_))
```

```
[0.5517475  0.39983413] 0.9515816
```




```
molwt = smiles_test.apply(Chem.MolFromSmiles).apply(Descriptors.MolMR)
plt.figure()
plt.scatter(red[:,0], red[:,1],marker='.', c= molwt)
```

```
<matplotlib.collections.PathCollection at 0x7f337e09a2e8>
```



Interpolating between molecules

```
def latent_to_smiles(latent):
    #decode states and set Reset the LSTM cells with them
    states = latent_to_states_model.predict(latent)
    sample_model.layers[1].reset_states(states=[states[0],states[1]])
    #Prepare the input char
    startidx = char_to_int["!"]
    samplevec = np.zeros((1,1,37))
    samplevec[0,0,1] = 1
    smiles = ""
    #Loop and predict next char
    for i in range(37):
        o = sample_model.predict(samplevec)
        sampleidx = np.argmax(o)
```

```
        samplechar = int_to_char[sampleidx]
        if samplechar != "E":
            smiles = smiles + int_to_char[sampleidx]
            samplevec = np.zeros((1,1,37))
            samplevec[0,0,sampleidx] = 1
        else:
            break
    return smiles

smiles = latent_to_smiles(x_latent[0:1])
```

```
print(smiles)
print(smiles_test.iloc[0])
```

```
C[NH+]1CCC[C@@H]1C(=O)NCC(=O)Nc1ccc(C
C[NH+]1CCC(NC(=O)[C@H]2CCN(c3ccc(Cl)c(Cl)c3)C2=O)CC1
```

```
i = 0
j= 2
latent1 = x_latent[j:j+1]
latent0 = x_latent[i:i+1]
mols1 = []
ratios = np.linspace(0,1,25)
for r in ratios:
    rlatent = (1.0-r)*latent0 + r*latent1
    smiles  = latent_to_smiles(rlatent)
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mols1.append(mol)
    else:
        print(smiles)
```

```
C[NH+]1CCC[C@@H]1C(=O)NCC(=O)Nc1ccc(C
COCCn1c(C)cc(NC(=O)N[C@@H](C)c2ccc(C(
COC(=O)c1ccc(NC(=O)C(=O)N2CCC[C@@H](C
COc1ccc(N2CCN(C(=O)C(=O)Nc3ccc(C(C)C)
O=C(NC1CCCC1)[C@@H]1CCCN(C(=O)c2ccc(C
O=C(NC1CCCC1)[C@@H]1CCCN(C(=O)c2ccc(C
O=C(NCC1(c2ccccc2)CCCC1)N1CCN(C(=O)c2
O=C(CCC1CCCC1)N1CCC[C@@H](C(=O)N2CCC[
O=C(Cc1ccccc1)N1CCC[C@@H](C(=O)N2CCC[
O=c1[nH]c(C(=O)N2CCC[C@@H]2c2ccccc2)c
N#C[C@@H](NC(=O)c2ccccc2Cl)C1(Cl)C(=O
N#C[C@@H](c2ccccc2)N1C(=O)C[NH+]1CCC[
NC(=O)c1ccc(CNC(=O)N2CCC[C@@H]2c2ccc(
NC(=O)COc1ccc(C(=O)N2CCC[C@@H]2c2ccc(
Cc1noc(CNC(=O)[C@@H]2CCCN(C(=O)c3ccc(
Cc1n[nH]c1CCC(=O)N1CCC[C@@H](C(=O)Nc2
Cc1n[nH]c1CCC(=O)N1CCC[C@@H](C(=O)Nc2
Cc1nc(CCC(=O)NC[C@@H](C)C(=O)Nc2ccc(C
Cc1c(C(=O)NCCc3ccccc3)ccc2[C@H](C)N(C
Cc1cc(C)c(C(=O)NCCC[NH+](C)Cc3ccccc3)
Cc1cc(C(=O)Nc3ccc(C(=O)NCc4ccccc4)cc3
Cc1cc(C(=O)Nc3ccc(C(=O)NCc4ccccc4)cc3
Cc1cc([C@H](C(=O)Nc4ccc(Cl)cc4)CC3)c(
Cc1ccc(C(=O)C(C)(C)C(=O)N2CCN(C)Cc2cc
Cc1ccc(C(=O)C(C)(C)C(=O)N2CCN(C)C(=O)
```

```
len(smiles)
```

```
37
```

Sample around latent space to find molecules that can be plotted

```
latent = x_latent[0:1]
scale = 0.40
mols = []
for i in range(20):
    latent_r = latent + scale*(np.random.randn(latent.shape[1])) #TODO, try with
    smiles = latent_to_smiles(latent_r)
    mol = Chem.MolFromSmiles(smiles)
    if mol:
        mols.append(mol)
    else:
        print(smiles)
```

```
C[NH+](CCC(=O)N1CCCC1)C(=O)N1CCC[C@@H
C[C@H](NC(=O)N[C@@H]1CCC[C@@H]1C(=O)[
C[NH+](C)CCNC(=O)N1CCC[C@@H](C(=O)Nc2
C[NH+]1CCC[C@@H]1C(=O)NCC(=O)Nc1ccc(C
C[C@H](NC(=O)N[C@@H]1CCC[C@@H]1C(=O)N
C[C@H](NC(=O)NC[C@@H]1CCCO1)c1ccc(C(=
C[NH+](C)CCNC(=O)N1CCC[C@@H](C(=O)Nc2
C[C@H](NC(=O)N[C@@H]1CCC[C@@H]1C(=O)[
C[NH+](CCC(=O)N1CCCC1)C(=O)N1CCC[C@@H
C[NH+](CCC(=O)N1CCCC1)C(=O)N1CCC[C@@H
C[NH+](CCC(=O)N1CCCC1)C(=O)N1CCC[C@@H
C[NH+](C)CCNC(=O)N1CCC[C@@H](C(=O)Nc2
C[C@H](NC(=O)N1CCC[C@H](C(=O)Nc2ccccc
C[C@H](NC(=O)N[C@@H]1CCC[C@@H]1C(=O)[
C[C@H](C(=O)N1CCCCC1)N1CCN(C(=O)c2ccc
C[C@H](NC(=O)NC[C@@H]1CCCO1)[C@@H]1CC
C[C@H](CC(=O)N1CCCCC1)[C@@H](C)[NH+]1
C[NH+](C)CCNC(=O)N1CCC[C@@H](C(=O)Nc2
```

```
mols
```

```
[<rdkit.Chem.rdchem.Mol at 0x7f3022c72710>,
 <rdkit.Chem.rdchem.Mol at 0x7f3022c72cb0>]
```

```
Draw.MolsToGridImage(mols, molsPerRow=5)
```