final

December 12, 2024

0.1 Abstract

0.2 Package import

[1]:
```python
import os

os.environ["KERAS_BACKEND"] = "tensorflow"

import ast
import numpy as np

from tensorflow import keras
```

[2]:
```python
#from tensorflow.keras import ops
from tensorflow.keras import layers
import pandas as pd

from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from rdkit import Chem, RDLogger
from rdkit.Chem import BondType
from rdkit.Chem.Draw import MolsToGridImage
from rdkit.Chem import Draw
from rdkit import Chem
from rdkit.Chem import rdmolops, AllChem
from tensorflow.keras.regularizers import l1_l2
RDLogger.DisableLog("rdApp.*")
```

[3]:
```python
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
print("GPU available:", tf.config.list_physical_devices('GPU'))
print("GPU in use:", tf.test.gpu_device_name())
```

```
TensorFlow version: 2.16.2
GPU available: []
GPU in use:
```

0.3 Database pharsing

[4]:
```python
'''
read the entire dataset
'''

df = pd.read_csv('dataset1.csv')
df.drop([0,1,2,3,4], inplace=True)
df=df.rename(columns = {'PUBCHEM_EXT_DATASOURCE_SMILES':
 ↪'SMILES','PUBCHEM_ACTIVITY_OUTCOME':'Activity', 'PUBCHEM_ACTIVITY_SCORE':
 ↪'Score'})
columns_to_drop = [col for col in df.columns if col not in ['SMILES',␣
 ↪'Activity', 'Score', 'Potency', 'Efficacy']]
df = df.drop(columns = columns_to_drop)
#df=df.drop(['Unnamed: 3','Unnamed: 4','Unnamed: 5'], axis=1)
df = df.dropna(subset=['SMILES'])

df=df.fillna(0)
print(df.head())
print(df.info())
```

```
                                              SMILES  Activity  Score  \
5          CNCC1=NC2=C(C=C(C=C2)Cl)C(=N1)C3=CC=CN3  Inactive    0.0
6                CCSC(=NC1=CC=C(C=C1)C(F)(F)F)N.Cl  Inactive    0.0
7  CCN(CC1=CC(=CC=C1)S(=O)(=O)[O-])C2=CC=C(C=C2)C…  Inactive    0.0
8  CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0
9  CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0

   Potency  Efficacy
5      0.0       0.0
6      0.0       0.0
7      0.0       0.0
8      0.0       0.0
9      0.0       0.0
<class 'pandas.core.frame.DataFrame'>
Index: 342051 entries, 5 to 342072
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   SMILES    342051 non-null  object
 1   Activity  342051 non-null  object
 2   Score     342051 non-null  float64
 3   Potency   342051 non-null  float64
 4   Efficacy  342051 non-null  float64
dtypes: float64(3), object(2)
memory usage: 15.7+ MB
None
```

[5]:
```
valid_indices = []
# Loop through each SMILES string in the DataFrame
for i in range(len(df)):
    smiles = df.iloc[i]['SMILES']  # Use iloc for positional indexing

    # Convert SMILES to molecule
    mol = Chem.MolFromSmiles(smiles)

    # Check if the molecule is valid and has <= 50 atoms
    if mol is not None and mol.GetNumAtoms() <= 50:
        valid_indices.append(i)
# Filter the DataFrame to include only valid molecules
df_50 = df.iloc[valid_indices]
```

[6]:
```
df_50
```

[6]:
```
                                                  SMILES  Activity  Score  \
5                    CNCC1=NC2=C(C=C(C=C2)Cl)C(=N1)C3=CC=CN3  Inactive    0.0
6                           CCSC(=NC1=CC=C(C=C1)C(F)(F)F)N.Cl  Inactive    0.0
8       CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0
9       CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0
10      C1CN(CCN1C2=NC(=NC3=CC=CC=C32)C4=CC=CS4)S(=O)(…  Inactive    0.0
…                                                    …       …    …
342068  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342069  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342070  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342071  CC(=O)NC1=CC=C(C=C1)C(=O)N(CC2=CC=CC=C2)CC3=CC…  Inactive    0.0
342072  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0

        Potency  Efficacy
5           0.0       0.0
6           0.0       0.0
8           0.0       0.0
9           0.0       0.0
10          0.0       0.0
…           …       …
342068      0.0       0.0
342069      0.0       0.0
342070      0.0       0.0
342071      0.0       0.0
342072      0.0       0.0

[341260 rows x 5 columns]
```

[7]:
```
def is_charged(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if not mol:
        return False  # Invalid SMILES
    return any(atom.GetFormalCharge() != 0 for atom in mol.GetAtoms())

# Test the function
print(is_charged("CC1=C(SC(=C1C#N)NC(=O)C2=CC(C=C2)OC)[N+](=O)"))
```

```
True
```

[8]:
```
df_50['Charged'] = df_50['SMILES'].apply(is_charged)

uncharged = df_50[df_50['Charged'] == False]
uncharged
```

```
/var/folders/jn/kkchdcr94t50xrmycsvkq2x80000gn/T/ipykernel_85584/162626946.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_50['Charged'] = df_50['SMILES'].apply(is_charged)
```

[8]:
```
                                                      SMILES  Activity  Score  \
5                     CNCC1=NC2=C(C=C(C=C2)Cl)C(=N1)C3=CC=CN3  Inactive    0.0
6                           CCSC(=NC1=CC=C(C=C1)C(F)(F)F)N.Cl  Inactive    0.0
8       CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0
9       CC1=CC=C(C=C1)S(=O)(=O)N2CCN(CC2)C3=NC(=NC4=CC…  Inactive    0.0
10      C1CN(CCN1C2=NC(=NC3=CC=CC=C32)C4=CC=CS4)S(=O)(…  Inactive    0.0
…                                                     …       …    …
342068  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342069  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342070  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0
342071  CC(=O)NC1=CC=C(C=C1)C(=O)N(CC2=CC=CC=C2)CC3=CC…  Inactive    0.0
342072  CC(=O)NC1=CC=C(C=C1)OCC2=C(C=CC(=C2)CN(CC3=CC=…  Inactive    0.0

        Potency  Efficacy  Charged
5           0.0       0.0    False
6           0.0       0.0    False
8           0.0       0.0    False
9           0.0       0.0    False
10          0.0       0.0    False
…         …       …       …
342068      0.0       0.0    False
342069      0.0       0.0    False
342070      0.0       0.0    False
342071      0.0       0.0    False
342072      0.0       0.0    False
```

```
[322199 rows x 6 columns]
```

[9]:
```
# Picking all "Active" molecules from the dataset
active_df = uncharged[uncharged['Activity'] == 'Active']
active_df.info()

# Picking all "Inactive" molecules from the dataset
inactive_df = uncharged[uncharged['Activity'] == 'Inactive']
inactive_df.info()

# Randomly sample from inactive_df to match the size of active_df
inactive_sampled = inactive_df.sample(n=len(active_df), random_state=42)

# Combine the active and sampled inactive molecules
balanced_df = pd.concat([active_df, inactive_sampled])

# Shuffle the combined dataset
balanced_df = balanced_df.sample(frac=1, random_state=42).reset_index(drop=True)

balanced_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6273 entries, 13 to 341825
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   SMILES    6273 non-null   object
 1   Activity  6273 non-null   object
 2   Score     6273 non-null   float64
 3   Potency   6273 non-null   float64
 4   Efficacy  6273 non-null   float64
 5   Charged   6273 non-null   bool
dtypes: bool(1), float64(3), object(2)
memory usage: 300.2+ KB
<class 'pandas.core.frame.DataFrame'>
Index: 304069 entries, 5 to 342072
Data columns (total 6 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   SMILES    304069 non-null  object
 1   Activity  304069 non-null  object
 2   Score     304069 non-null  float64
 3   Potency   304069 non-null  float64
 4   Efficacy  304069 non-null  float64
 5   Charged   304069 non-null  bool
dtypes: bool(1), float64(3), object(2)
memory usage: 14.2+ MB
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 12546 entries, 0 to 12545
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   SMILES    12546 non-null  object
 1   Activity  12546 non-null  object
 2   Score     12546 non-null  float64
 3   Potency   12546 non-null  float64
 4   Efficacy  12546 non-null  float64
 5   Charged   12546 non-null  bool
dtypes: bool(1), float64(3), object(2)
memory usage: 502.5+ KB
```

[10]:
```
filtered_df = balanced_df
filtered_df
```

[10]:
```
                                                  SMILES  Activity  Score  \
0             CC1=C(C=CC=C1Br)NC(=O)C2=C(C=CS2)N3C=CC=C3    Active   82.0
1                      CCCCCC(C(C)CC(=O)NC1CCCCC1)C(=O)O    Active   43.0
2      CC1=CC=C(C=C1)S(=O)(=O)NC2=NN3C(C=C(NC3=N2)C)C…    Active   41.0
3      CC1=CC(=O)OC2=C1C=C(C=C2)OCC(=O)NC3=CC=CC(=C3)…  Inactive    0.0
4      CC1=CC(=C(N1C)C)C(=O)COC(=O)C23CC4CC(C2)CC(C4)…  Inactive    0.0
…                                                    …       …    …
12541  C1CN(CCN1C(=O)C2=CC=CC=C2CC3=CC=CC=C3)S(=O)(=O…  Inactive    0.0
12542                C1=CC=C(C=C1)OC2=NC=NC(=C2)N3C=NC=N3    Active   64.0
12543  CC1=C(C(=CC=C1)N2CCN(CC2)C3=NC4=CC=CC=C4C(=O)N…    Active   42.0
12544  CCC(C)NC(=O)CSC1=NC2=CC=CC=C2C3=NC(C(=O)N31)C4…    Active   42.0
12545                  CC(C)C1=CC=C(C=C1)S(=O)(=O)NC2CCCC2  Inactive    0.0

       Potency  Efficacy  Charged
0       8.9125  140.7280    False
1      12.5893  136.6590    False
2      22.3872  166.6580    False
3       0.0000    0.0000    False
4       0.0000    0.0000    False
…          …         …      …
12541   0.0000    0.0000    False
12542   2.8184   74.9734    False
12543  17.7828  126.5240    False
12544  15.8489  139.3040    False
12545   0.0000    0.0000    False

[12546 rows x 6 columns]
```

0.4 Parameter setting

[11]:
```
'''
scan through all the molecules to obtain unique atom types
'''
smiles = filtered_df['SMILES'].tolist()
search_elements=[]
for smile in smiles:
    mol = Chem.MolFromSmiles(smile)
    atoms = list(set([atom.GetSymbol() for atom in mol.GetAtoms()]))
    search_elements += atoms
    search_elements = list(set(search_elements))
search_elements.append("H")
print(search_elements)
```

```
['C', 'F', 'N', 'I', 'O', 'P', 'Br', 'B', 'S', 'Cl', 'As', 'H']
```

[12]:
```
'''
Setting up the atom mapping and bond mapping.
Code adopted from https://keras.io/examples/generative/molecule_generation/
'''
SMILE_CHARSET = str(search_elements)
bond_mapping = {"SINGLE": 0, "DOUBLE": 1, "TRIPLE": 2, "AROMATIC": 3}
bond_mapping.update(
    {0: BondType.SINGLE, 1: BondType.DOUBLE, 2: BondType.TRIPLE, 3: BondType.
 ↪AROMATIC}
)
SMILE_CHARSET = ast.literal_eval(SMILE_CHARSET)

MAX_MOLSIZE = max(filtered_df['SMILES'].str.len())
SMILE_to_index = dict((c, i) for i, c in enumerate(SMILE_CHARSET))
index_to_SMILE = dict((i, c) for i, c in enumerate(SMILE_CHARSET))
atom_mapping = dict(SMILE_to_index)
atom_mapping.update(index_to_SMILE)
print(atom_mapping)
print("Max molecule size: {}".format(MAX_MOLSIZE))
print("Character set Length: {}".format(len(SMILE_CHARSET)))
```

```
{'C': 0, 'F': 1, 'N': 2, 'I': 3, 'O': 4, 'P': 5, 'Br': 6, 'B': 7, 'S': 8, 'Cl':
9, 'As': 10, 'H': 11, 0: 'C', 1: 'F', 2: 'N', 3: 'I', 4: 'O', 5: 'P', 6: 'Br',
7: 'B', 8: 'S', 9: 'Cl', 10: 'As', 11: 'H'}
Max molecule size: 117
Character set Length: 12
```

0.5 Hyperparameters

[13]:
```
'''
Defining the Hyperparameters of the model
'''

NUM_ATOMS = 50 #Max number of atoms
ATOM_DIM = len(SMILE_CHARSET)  # Number of atom types
BOND_DIM = 5 # Number of bond types
```

0.6 Molecule featurization

[14]:
```
'''
Defining functions to convert smiles string into node graph and recover␣
 ↪molecule structure from it.
Code referenced from: https://keras.io/examples/generative/molecule_generation/
'''


def smiles_to_graph(smiles):
    '''
    Reference: https://keras.io/examples/generative/wgan-graphs/
    '''
    # Converts SMILES to molecule object
    molecule = Chem.MolFromSmiles(smiles)
    #molecule = Chem.AddHs(molecule)
    # Initialize adjacency and feature tensor
    adjacency = np.zeros((BOND_DIM, NUM_ATOMS, NUM_ATOMS), "float32")
    features = np.zeros((NUM_ATOMS, ATOM_DIM), "float32")

    # loop over each atom in molecule
    for atom in molecule.GetAtoms():
        i = atom.GetIdx()
        atom_type = atom_mapping[atom.GetSymbol()]
        features[i] = np.eye(ATOM_DIM)[atom_type]
        # loop over one-hop neighbors
        for neighbor in atom.GetNeighbors():
            j = neighbor.GetIdx()
            bond = molecule.GetBondBetweenAtoms(i, j)
            bond_type_idx = bond_mapping[bond.GetBondType().name]
            adjacency[bond_type_idx, [i, j], [j, i]] = 1

    # Where no bond, add 1 to last channel (indicating "non-bond")
    # Notice: channels-first
    adjacency[-1, np.sum(adjacency, axis=0) == 0] = 1

    # Where no atom, add 1 to last column (indicating "non-atom")
    features[np.where(np.sum(features, axis=1) == 0)[0], -1] = 1

    return adjacency, features

def graph_to_molecule(adjacency, features):
    # RWMol is a molecule object intended to be edited
    molecule = Chem.RWMol()
    # Remove "no atoms" & atoms with no bonds
    keep_idx = np.where(
        (np.argmax(features, axis=1) != ATOM_DIM - 1)
        & (np.sum(adjacency[:-1], axis=(0, 1)) > 0))[0]

    features = features[keep_idx]
    adjacency = adjacency[:, keep_idx][:, :, keep_idx]

    # Add atoms to molecule
    for atom_type_idx in np.argmax(features, axis=1):
        atom = Chem.Atom(atom_mapping[atom_type_idx])
        _ = molecule.AddAtom(atom)

    added_bonds = set()
    (bonds_ij, atoms_i, atoms_j) = np.where(np.triu(adjacency) == 1)
    for (bond_ij, atom_i, atom_j) in zip(bonds_ij, atoms_i, atoms_j):
        if atom_i == atom_j or bond_ij == BOND_DIM - 1:
            continue
        bond_type = bond_mapping.get(bond_ij, None)
        if (atom_i, atom_j) in added_bonds or (atom_j, atom_i) in added_bonds:

            continue
        molecule.AddBond(int(atom_i), int(atom_j), bond_type)
        added_bonds.add((atom_i, atom_j))


     # Sanitize without Kekulization
    try:
        Chem.SanitizeMol(molecule, sanitizeOps=Chem.SanitizeFlags.SANITIZE_ALL␣
 ↪^ Chem.SanitizeFlags.SANITIZE_KEKULIZE)
    except Exception as e:
        print(f"Sanitization failed: {e}")
        return None

    # Add explicit hydrogens
    molecule_with_h = Chem.AddHs(molecule)

    # Fix aromaticity in aromatic rings
    for atom in molecule_with_h.GetAtoms():
        if atom.GetIsAromatic():
            atom.SetIsAromatic(False)  # Clear aromaticity if needed

    # Force Kekulization to alternate bond orders in aromatic rings
    try:
        Chem.Kekulize(molecule_with_h, clearAromaticFlags=True)
    except Chem.KekulizeException as e:
        print(f"Kekulization failed: {e}")
        return molecule_with_h  # Return molecule without Kekulé bonds

    return molecule_with_h
```

0.7 Building model

[15]:
```
'''
    Defining GCN
    Reference: https://keras.io/examples/generative/wgan-graphs/
    The Encoder takes as input a molecule's graph adjacency matrix and feature␣
 ↪matrix.
'''
class RelationalGraphConvLayer(keras.layers.Layer):
    def __init__(
        self,
        units=128,
        activation="relu",
        use_bias=False,
        kernel_initializer="glorot_uniform",
        bias_initializer="zeros",
        kernel_regularizer=None,
        bias_regularizer=None,
        **kwargs
    ):
        super().__init__(**kwargs)

        self.units = units
        self.activation = keras.activations.get(activation)
        self.use_bias = use_bias
        self.kernel_initializer = keras.initializers.get(kernel_initializer)
        self.bias_initializer = keras.initializers.get(bias_initializer)
        self.kernel_regularizer = keras.regularizers.get(kernel_regularizer)
        self.bias_regularizer = keras.regularizers.get(bias_regularizer)

    def build(self, input_shape):
        bond_dim = input_shape[0][1]
        atom_dim = input_shape[1][2]

        self.kernel = self.add_weight(
            shape=(bond_dim, atom_dim, self.units),
            initializer=self.kernel_initializer,
            regularizer=self.kernel_regularizer,
            trainable=True,
            name="W",
            dtype=tf.float32,
        )

        if self.use_bias:
            self.bias = self.add_weight(
                shape=(bond_dim, 1, self.units),
                initializer=self.bias_initializer,
                regularizer=self.bias_regularizer,
                trainable=True,
                name="b",
                dtype=tf.float32,
            )

        self.built = True

    def call(self, inputs, training=False):
        adjacency, features = inputs
        # Aggregate information from neighbors
        x = tf.matmul(adjacency, features[:, None, :, :])
        # Apply linear transformation
        x = tf.matmul(x, self.kernel)
        if self.use_bias:
            x += self.bias
        # Reduce bond types dim
        x_reduced = tf.reduce_sum(x, axis=1)
        # Apply non-linear transformation
        return self.activation(x_reduced)
```

0.8 Build the Encoder and Decoder

[16]:
```
'''
defining function to build encoder and decoder.
Code adopted and modified from https://keras.io/examples/generative/
 ↪molecule_generation/
'''

def get_encoder(gconv_units, latent_dim, adjacency_shape, feature_shape,␣
 ↪dense_units, dropout_rate, regularizer=None):
    adjacency = keras.layers.Input(shape=adjacency_shape,␣
 ↪name="adjacency_input")
    features = keras.layers.Input(shape=feature_shape, name="feature_input")
    scores = keras.layers.Input(shape=(1,), name="score_input")  # Conditional␣
 ↪input (scalar)
```

```
    # Graph convolution layers
    features_transformed = features
    for units in gconv_units:
        features_transformed = RelationalGraphConvLayer(units)(
            [adjacency, features_transformed]
        )

    # Reduce 2D representation to 1D
    x = keras.layers.GlobalAveragePooling1D()(features_transformed)

    # Concatenate the score (condition) to the reduced graph representation
    x = keras.layers.Concatenate()([x, scores])

    # Fully connected layers
    for units in dense_units:
        x = layers.Dense(units, activation="relu",␣
 ↪kernel_regularizer=regularizer)(x)
        x = layers.Dropout(dropout_rate)(x)

    # Latent space
    z_mean = layers.Dense(latent_dim, name="z_mean")(x)
    z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)

    # Create encoder model
    encoder = keras.Model(inputs=[adjacency, features, scores],␣
 ↪outputs=[z_mean, z_log_var], name="encoder")
    encoder.summary()
    return encoder


class SymmetrizeLayer(layers.Layer):
    def call(self, x):
        return (x + tf.transpose(x, (0, 1, 3, 2))) / 2

def get_decoder(dense_units, latent_dim, adjacency_shape, feature_shape,␣
 ↪dropout_rate, regularizer=None):
    latent_input = keras.Input(shape=(latent_dim,), name="latent_input")
    scores = keras.Input(shape=(1,), name="score_input")  # Conditional input␣
 ↪(scalar)

    # Concatenate latent input with the conditional score
    x = keras.layers.Concatenate()([latent_input, scores])

    # Dense layers
    for units in dense_units:
        x = keras.layers.Dense(units, activation="tanh",␣
↪kernel_regularizer=regularizer)(x)
        x = keras.layers.Dropout(dropout_rate)(x)

    # Adjacency reconstruction
    adj_output = keras.layers.Dense(tf.math.reduce_prod(adjacency_shape).
↪numpy().astype(int))(x)
    adj_output = keras.layers.Reshape(adjacency_shape)(adj_output)
    adj_output = SymmetrizeLayer()(adj_output)
    adj_output = keras.layers.Softmax(axis=1)(adj_output)

    # Feature reconstruction
    feat_output = keras.layers.Dense(tf.math.reduce_prod(feature_shape).numpy().
↪astype(int))(x)
    feat_output = keras.layers.Reshape(feature_shape)(feat_output)
    feat_output = keras.layers.Softmax(axis=2)(feat_output)

    # Create decoder model
    decoder = keras.Model(inputs=[latent_input, scores], outputs=[adj_output,␣
↪feat_output], name="decoder")
    decoder.summary()
    return decoder
```

0.9 Build the VAE

[17]:
```
'''
defining the VAE
Code adopted and modified from https://keras.io/examples/generative/
↪molecule_generation/
'''

class VAE(keras.Model):
    def __init__(self, encoder, decoder, beta=1.0, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder
        self.beta = beta

    def call(self, inputs):
        adjacency, features, scores = inputs
        z_mean, z_log_var = self.encoder([adjacency, features, scores])
        z = self.reparameterize(z_mean, z_log_var)
        return self.decoder([z, scores])
    def sampling(self, args):
        """
        Reparameterization trick: Sample from a Gaussian distribution using
        z = z_mean + epsilon * exp(z_log_var / 2), where epsilon is sampled␣
↪from N(0, 1).
        """
        z_mean, z_log_var = args
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.keras.backend.random_normal(shape=(batch, dim))  #␣
↪Standard normal noise
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon
```

0.10 Model training

[18]:
```
'''
splitting the dataset into training and testing
'''

train, test = train_test_split(filtered_df,test_size=0.2,random_state=42)
train_df, val_df = train_test_split(train, test_size=0.2, random_state=42)
train_df.reset_index(drop=True, inplace=True)
val_df.reset_index(drop=True, inplace=True)
test.reset_index(drop=True, inplace=True)

adj_train, fea_train, score_train = [], [], []
adj_val, fea_val, score_val = [], [], []

for idx in range(len(train_df)):
    adjacency, features = smiles_to_graph(train_df.loc[idx]["SMILES"])
    score = train_df.loc[idx]["Score"]
    adj_train.append(adjacency)
    fea_train.append(features)
    score_train.append(score)

for idx in range(len(val_df)):
    adjacency, features = smiles_to_graph(val_df.loc[idx]["SMILES"])
    score = val_df.loc[idx]["Score"]
    adj_val.append(adjacency)
    fea_val.append(features)
    score_val.append(score)


adj_train = np.array(adj_train)
fea_train = np.array(fea_train)
score_train_ = np.array(score_train).reshape(-1,1)

adj_val = np.array(adj_val)
fea_val = np.array(fea_val)
score_val_ = np.array(score_val).reshape(-1,1)
```

[19]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

score_train_n = scaler.fit_transform(score_train_)
score_val_n = scaler.transform(score_val_)
```

[20]:
```
print(adj_train.shape)
print(fea_train.shape)
print(score_train_.shape)
print(adj_val.shape)
print(fea_val.shape)
print(score_val_.shape)
```

```
(8028, 5, 50, 50)
(8028, 50, 12)
(8028, 1)
(2008, 5, 50, 50)
(2008, 50, 12)
(2008, 1)
```

[21]:
```
print(np.max(score_train_n))
```

```
0.9999999999999999
```

[22]:
```
#Hyperparameters
BATCH_SIZE = 64
EPOCHS = 25
VAE_LR = 3e-4 # changed to 1e-3
LATENT_DIM = 128  # Size of the latent space
```

[23]:
```
'''
compiling the VAE
'''

encoder = get_encoder(
    gconv_units=[16],
    adjacency_shape=(BOND_DIM, NUM_ATOMS, NUM_ATOMS),
    feature_shape=(NUM_ATOMS, ATOM_DIM),
    latent_dim=LATENT_DIM,
    dense_units=[256, 512],
    dropout_rate=0,
    regularizer=l1_l2(l1=1e-6, l2=1e-3)
)
decoder = get_decoder(
    dense_units=[128, 256, 512],
    dropout_rate=0.3,
    latent_dim=LATENT_DIM,
    adjacency_shape=(BOND_DIM, NUM_ATOMS, NUM_ATOMS),
    feature_shape=(NUM_ATOMS, ATOM_DIM),
    regularizer=l1_l2(l1=1e-4, l2=1e-2)
)
vae = VAE(encoder, decoder)

vae.compile(optimizer=keras.optimizers.Adam(learning_rate=VAE_LR))
```

```
Model: "encoder"


 Layer (type)           Output Shape          Param #  Connected to

 adjacency_input        (None, 5, 50, 50)           0  -
 (InputLayer)

 feature_input          (None, 50, 12)              0  -
 (InputLayer)

 relational_graph_c…    (None, 50, 16)            960  adjacency_input[…
 (RelationalGraphCo…                                   feature_input[0]…

 global_average_poo…    (None, 16)                  0  relational_graph…
 (GlobalAveragePool…

 score_input            (None, 1)                   0  -
 (InputLayer)

 concatenate            (None, 17)                  0  global_average_p…
 (Concatenate)                                         score_input[0][0]

 dense (Dense)          (None, 256)             4,608  concatenate[0][0]

 dropout (Dropout)      (None, 256)                 0  dense[0][0]

 dense_1 (Dense)        (None, 512)           131,584  dropout[0][0]

 dropout_1 (Dropout)    (None, 512)                 0  dense_1[0][0]

 z_mean (Dense)         (None, 128)            65,664  dropout_1[0][0]

 z_log_var (Dense)      (None, 128)            65,664  dropout_1[0][0]


 Total params: 268,480 (1.02 MB)
 Trainable params: 268,480 (1.02 MB)

 Non-trainable params: 0 (0.00 B)

Model: "decoder"


 Layer (type)          Output Shape             Param #   Connected to

 latent_input          (None, 128)                    0   -
 (InputLayer)

 score_input           (None, 1)                      0   -
 (InputLayer)

 concatenate_1         (None, 129)                    0   latent_input[0][…
 (Concatenate)                                            score_input[0][0]

 dense_2 (Dense)       (None, 128)               16,640   concatenate_1[0]…

 dropout_2 (Dropout)   (None, 128)                    0   dense_2[0][0]

 dense_3 (Dense)       (None, 256)               33,024   dropout_2[0][0]

 dropout_3 (Dropout)   (None, 256)                    0   dense_3[0][0]

 dense_4 (Dense)       (None, 512)              131,584   dropout_3[0][0]

 dropout_4 (Dropout)   (None, 512)                    0   dense_4[0][0]

 dense_5 (Dense)       (None, 12500)          6,412,500   dropout_4[0][0]

 reshape (Reshape)     (None, 5, 50, 50)              0   dense_5[0][0]

 dense_6 (Dense)       (None, 600)              307,800   dropout_4[0][0]

 symmetrize_layer      (None, 5, 50, 50)              0   reshape[0][0]
 (SymmetrizeLayer)

 reshape_1 (Reshape)   (None, 50, 12)                 0   dense_6[0][0]

 softmax (Softmax)     (None, 5, 50, 50)              0   symmetrize_layer…

 softmax_1 (Softmax)   (None, 50, 12)                 0   reshape_1[0][0]
```

```
Total params: 6,901,548 (26.33 MB)

Trainable params: 6,901,548 (26.33 MB)

Non-trainable params: 0 (0.00 B)
```

[24]:
```
val_loss_list = []
train_loss_list = []
kl_theshold = 1.0
```

[25]:
```
train_dataset = tf.data.Dataset.from_tensor_slices((adj_train, fea_train,␣
 ↪score_train_)).batch(BATCH_SIZE)
val_dataset = tf.data.Dataset.from_tensor_slices((adj_val, fea_val,␣
 ↪score_val_)).batch(BATCH_SIZE)
```

[26]:
```
for epoch in range(EPOCHS):
    print(f"Epoch {epoch + 1}/{EPOCHS}")
    if epoch < 10:
        beta = 0.05
    else:
        beta = epoch*0.01
    # Training Loop
    train_loss = 0
    for (adjacency, features, scores) in train_dataset:
        with tf.GradientTape() as tape:
            # Forward pass
            z_mean, z_log_var = vae.encoder([adjacency, features, scores])
            z = vae.sampling([z_mean, z_log_var])
            adj_reconstruction, feature_reconstruction = vae.decoder([z,␣
 ↪scores])

            # Compute losses
            adj_loss = tf.reduce_mean(
                tf.reduce_sum(keras.losses.binary_crossentropy(adjacency,␣
 ↪adj_reconstruction), axis=(1, 2))
            )
            feat_loss = tf.reduce_mean(
                tf.reduce_sum(keras.losses.categorical_crossentropy(features,␣
 ↪feature_reconstruction), axis=1)
            )
            reconstruction_loss = adj_loss + feat_loss
            kl_loss = -0.5 * tf.reduce_mean(
                tf.reduce_sum(1 + z_log_var - tf.square(z_mean) - tf.
 ↪exp(z_log_var), axis=1)
            )
            total_loss = reconstruction_loss + beta * kl_loss

        # Backpropagation
        grads = tape.gradient(total_loss, vae.trainable_weights)
        vae.optimizer.apply_gradients(zip(grads, vae.trainable_weights))

        train_loss += total_loss


    train_loss /= len(train_dataset)
    train_loss_list.append(train_loss)

    print(f"Train Loss: {train_loss.numpy()}, KL Loss: {kl_loss.numpy()},␣
↪Reconstruction Loss: {reconstruction_loss.numpy()}")

    # Validation Loop
    val_loss = 0
    for (val_adjacency, val_features, val_scores) in val_dataset:
        # Forward pass
        z_mean, z_log_var = vae.encoder([val_adjacency, val_features,␣
↪val_scores])
        z = vae.sampling([z_mean, z_log_var])
        val_adj_reconstruction, val_feat_reconstruction = vae.decoder([z,␣
↪val_scores])

        # Compute losses
        val_adj_loss = tf.reduce_mean(
            tf.reduce_sum(keras.losses.binary_crossentropy(val_adjacency,␣
↪val_adj_reconstruction), axis=(1, 2))
        )
        val_feat_loss = tf.reduce_mean(
            tf.reduce_sum(keras.losses.categorical_crossentropy(val_features,␣
↪val_feat_reconstruction), axis=1)
        )
        val_reconstruction_loss = val_adj_loss + val_feat_loss
        val_kl_loss = -0.5 * tf.reduce_mean(
            tf.reduce_sum(1 + z_log_var - tf.square(z_mean) - tf.
↪exp(z_log_var), axis=1)
        )
        val_total_loss = val_reconstruction_loss + beta * val_kl_loss

        val_loss += val_total_loss

    val_loss /= len(val_dataset)
    val_loss_list.append(val_loss)
    # Adjust beta if KL loss is very low
    if kl_loss < kl_theshold:
        beta = 0.05
    print(f"Validation Loss: {val_loss.numpy()}, KL Loss: {val_kl_loss.
↪numpy()}, Reconstruction Loss: {val_reconstruction_loss.numpy()}")
    print('BETA is: ', beta)
```

```
Epoch 1/25

2024-12-12 17:13:48.636583: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 66.95944213867188, KL Loss: 5.409961223602295, Reconstruction Loss:
36.86543655395508

2024-12-12 17:13:49.281854: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 38.307167053222656, KL Loss: 7.239017486572266, Reconstruction
Loss: 35.230045318603516
BETA is:  0.05
Epoch 2/25

2024-12-12 17:13:57.845969: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 38.08542251586914, KL Loss: 7.262415409088135, Reconstruction Loss:
35.81265640258789

2024-12-12 17:13:58.369698: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 37.226993560791016, KL Loss: 8.832179069519043, Reconstruction
Loss: 34.20383071899414
BETA is:  0.05
Epoch 3/25

2024-12-12 17:14:07.423950: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 34.377647399902344, KL Loss: 13.722108840942383, Reconstruction
Loss: 31.014507293701172

2024-12-12 17:14:07.964127: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 32.02784729003906, KL Loss: 14.46497631072998, Reconstruction
Loss: 29.39027976989746
BETA is:  0.05
Epoch 4/25

2024-12-12 17:14:16.276468: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Train Loss: 31.324237823486328, KL Loss: 15.213322639465332, Reconstruction
Loss: 29.61969566345215

2024-12-12 17:14:16.802566: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.837430953979492, KL Loss: 15.234996795654297, Reconstruction
Loss: 27.43654441833496
BETA is:  0.05
Epoch 5/25

2024-12-12 17:14:24.980889: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.58317756652832, KL Loss: 14.706596374511719, Reconstruction Loss:
29.678884506225586

2024-12-12 17:14:25.507104: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.382816314697266, KL Loss: 16.113767623901367, Reconstruction
Loss: 27.395017623901367
BETA is:  0.05
Epoch 6/25

2024-12-12 17:14:33.694724: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.195568084716797, KL Loss: 15.516021728515625, Reconstruction
Loss: 28.656808853149414

2024-12-12 17:14:34.217531: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.03752899169922, KL Loss: 16.778310775756836, Reconstruction
Loss: 27.231407165527344
BETA is:  0.05
Epoch 7/25

2024-12-12 17:14:42.781334: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.25861930847168, KL Loss: 16.683490753173828, Reconstruction Loss:
28.643312454223633

2024-12-12 17:14:43.308068: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.948104858398438, KL Loss: 17.50771141052246, Reconstruction
Loss: 26.600330352783203
BETA is:  0.05
Epoch 8/25

2024-12-12 17:14:51.590812: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Train Loss: 29.655874252319336, KL Loss: 15.88936710357666, Reconstruction Loss:
28.086673736572266

2024-12-12 17:14:52.134219: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.685688018798828, KL Loss: 16.187108993530273, Reconstruction
Loss: 26.570858001708984
BETA is:  0.05
Epoch 9/25

2024-12-12 17:15:00.586432: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.42094612121582, KL Loss: 15.739115715026855, Reconstruction Loss:
28.297420501708984

2024-12-12 17:15:01.104570: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.49125862121582, KL Loss: 16.52348518371582, Reconstruction
Loss: 26.6302433013916
BETA is:  0.05
Epoch 10/25

2024-12-12 17:15:09.567166: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.231225967407227, KL Loss: 15.30255126953125, Reconstruction Loss:
27.82144546508789

2024-12-12 17:15:10.087751: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.279537200927734, KL Loss: 16.1846981048584, Reconstruction
Loss: 26.073829650878906
BETA is:  0.05
Epoch 11/25

2024-12-12 17:15:18.996780: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.884469985961914, KL Loss: 12.192437171936035, Reconstruction
Loss: 28.23736000061035

2024-12-12 17:15:19.550752: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.96446990966797, KL Loss: 13.384326934814453, Reconstruction
Loss: 26.216266632080078
BETA is:  0.1
Epoch 12/25

2024-12-12 17:15:27.985540: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Train Loss: 29.867341995239258, KL Loss: 10.197785377502441, Reconstruction
Loss: 28.79987907409668

2024-12-12 17:15:28.528380: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.928882598876953, KL Loss: 10.827570915222168, Reconstruction
Loss: 28.151025772094727
BETA is:  0.11
Epoch 13/25

2024-12-12 17:15:36.862064: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.78401756286621, KL Loss: 9.285300254821777, Reconstruction Loss:
28.00973892211914

2024-12-12 17:15:37.384640: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.772369384765625, KL Loss: 9.544482231140137, Reconstruction
Loss: 26.41633415222168
BETA is:  0.12
Epoch 14/25

2024-12-12 17:15:45.740881: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.474695205688477, KL Loss: 9.579269409179688, Reconstruction Loss:
28.297019958496094

2024-12-12 17:15:46.264128: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.842233657836914, KL Loss: 9.950072288513184, Reconstruction
Loss: 26.075977325439453
BETA is:  0.13
Epoch 15/25

2024-12-12 17:15:54.413123: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.662763595581055, KL Loss: 7.968265533447266, Reconstruction Loss:
27.789501190185547

2024-12-12 17:15:54.943067: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.198326110839844, KL Loss: 8.257922172546387, Reconstruction
Loss: 26.271183013916016
BETA is:  0.14
Epoch 16/25

2024-12-12 17:16:03.272911: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Train Loss: 30.350461959838867, KL Loss: 8.190351486206055, Reconstruction Loss:
28.928070068359375

2024-12-12 17:16:03.803879: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.776681900024414, KL Loss: 8.088484764099121, Reconstruction
Loss: 26.190954208374023
BETA is:  0.15
Epoch 17/25

2024-12-12 17:16:12.172904: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.108983993530273, KL Loss: 7.2249579429626465, Reconstruction
Loss: 27.830188751220703

2024-12-12 17:16:12.691355: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.608184814453125, KL Loss: 7.540590763092041, Reconstruction
Loss: 26.069196701049805
BETA is:  0.16
Epoch 18/25

2024-12-12 17:16:20.934304: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.48594856262207, KL Loss: 6.611556053161621, Reconstruction Loss:
27.594058990478516

2024-12-12 17:16:21.456986: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.519359588623047, KL Loss: 6.855264186859131, Reconstruction
Loss: 27.162221908569336
BETA is:  0.17
Epoch 19/25

2024-12-12 17:16:29.780382: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.709335327148438, KL Loss: 6.697078704833984, Reconstruction Loss:
27.527141571044922

2024-12-12 17:16:30.301756: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.652938842773438, KL Loss: 6.887571811676025, Reconstruction
Loss: 26.06425666809082
BETA is:  0.18
Epoch 20/25

2024-12-12 17:16:38.987568: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Train Loss: 30.029027938842773, KL Loss: 6.425731182098389, Reconstruction Loss:
27.666391372680664

2024-12-12 17:16:39.494257: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.664268493652344, KL Loss: 6.584661960601807, Reconstruction
Loss: 26.06829261779785
BETA is:  0.19
Epoch 21/25

2024-12-12 17:16:47.847076: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.68038558959961, KL Loss: 6.201575756072998, Reconstruction Loss:
27.925024032592773

2024-12-12 17:16:48.368611: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.514310836791992, KL Loss: 6.663267612457275, Reconstruction
Loss: 25.973363876342773
BETA is:  0.2
Epoch 22/25

2024-12-12 17:16:56.768037: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.492443084716797, KL Loss: 6.026613712310791, Reconstruction Loss:
27.56798553466797

2024-12-12 17:16:57.305910: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.594175338745117, KL Loss: 6.642364501953125, Reconstruction
Loss: 25.876943588256836
BETA is:  0.21
Epoch 23/25

2024-12-12 17:17:05.655918: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.715835571289062, KL Loss: 9.752555847167969, Reconstruction Loss:
27.58233642578125

2024-12-12 17:17:06.196618: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.531145095825195, KL Loss: 12.062098503112793, Reconstruction
Loss: 25.871355056762695
BETA is:  0.22
Epoch 24/25

2024-12-12 17:17:14.603124: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

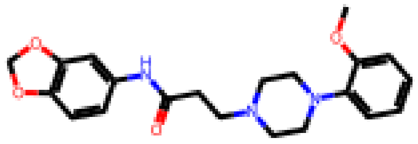
```
Train Loss: 29.64225196838379, KL Loss: 5.31318998336792, Reconstruction Loss:
27.660545349121094

2024-12-12 17:17:15.133052: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.5415096282959, KL Loss: 5.519102096557617, Reconstruction
Loss: 26.132801055908203
BETA is:  0.23
Epoch 25/25

2024-12-12 17:17:23.421472: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.305187225341797, KL Loss: 5.078959941864014, Reconstruction Loss:
27.42971420288086
Validation Loss: 29.36880874633789, KL Loss: 5.2280707359313965, Reconstruction
Loss: 25.697792053222656
BETA is:  0.24

2024-12-12 17:17:23.960953: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

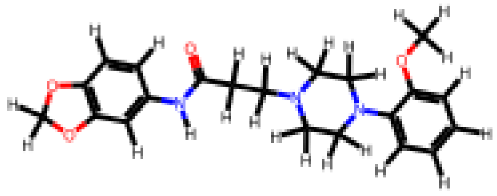
[27]:
```
'''
Checking the model's ability to reconstruct a molecule from the training dataset
'''

i=10
adjacency_check, features_check = smiles_to_graph(train_df.loc[i]["SMILES"])
score_check = [train_df.loc[i]["Score"]]
molobj = Chem.MolFromSmiles(train_df.loc[i]["SMILES"])
adj0 = np.expand_dims(adjacency_check,axis=0)
feature0 = np.expand_dims(features_check,axis=0)
score0 = np.expand_dims(score_check,axis=0)
print(adj0.shape)
print(feature0.shape)
print(score0.shape)
```

```
(1, 5, 50, 50)
(1, 50, 12)
(1, 1)
```

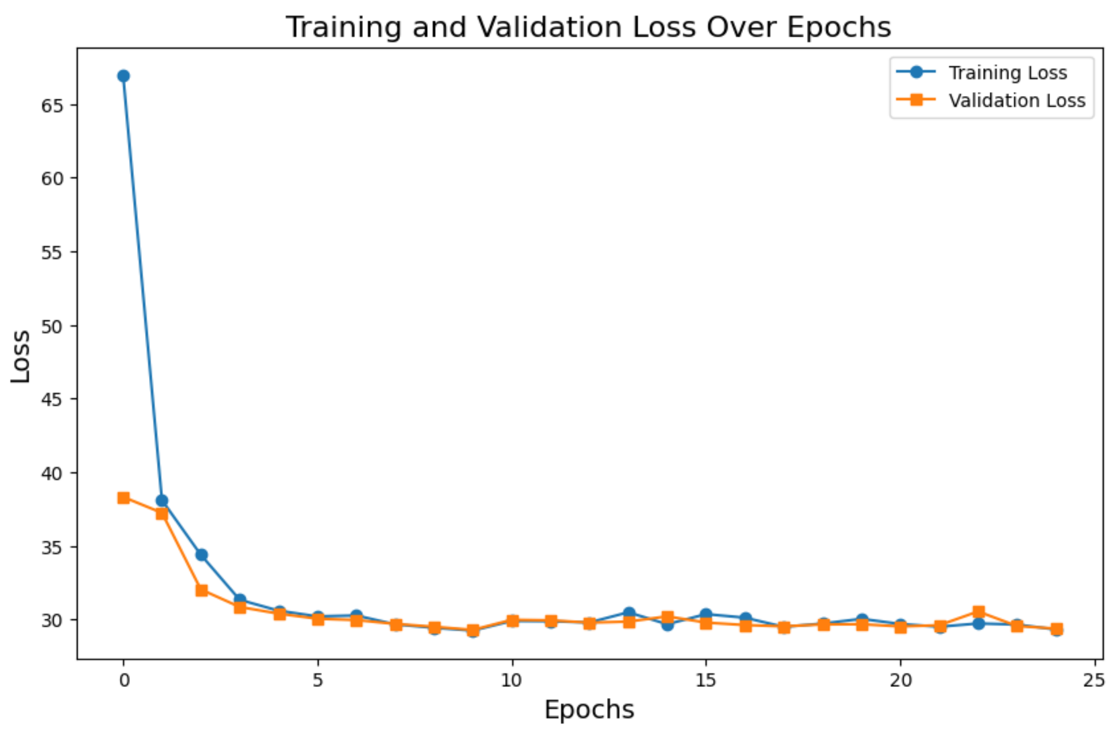
[28]:
```
mole_pred = graph_to_molecule(adj0[0], feature0[0])
Draw.MolsToGridImage([molobj,mole_pred], molsPerRow=2,)
```

[28]:





```
[29]: plt.figure(figsize=(10, 6))
      plt.plot(range(EPOCHS), train_loss_list, label='Training Loss', marker='o')
      plt.plot(range(EPOCHS), val_loss_list, label='Validation Loss', marker='s')

      # Add title and labels
      plt.title('Training and Validation Loss Over Epochs', fontsize=16)
      plt.xlabel('Epochs', fontsize=14)
      plt.ylabel('Loss', fontsize=14)
      plt.legend()
      plt.show()
```



0.11 Visualize latent space

[30]:
```
adj_test, fea_test, score_test = [], [], []

for idx in range(len(test)):
    adjacency, features = smiles_to_graph(test.loc[idx]["SMILES"])
    score = test.loc[idx]["Score"]
    adj_test.append(adjacency)
    fea_test.append(features)
    score_test.append(score)


adj_test = np.array(adj_test)
fea_test = np.array(fea_test)
score_test_ = np.array(score_test).reshape(-1,1)

score_test_n = scaler.transform(score_test_)
```

[31]:
```
ls_train = vae.encoder.predict([adj_train, fea_train, score_train_])
ls_test = vae.encoder.predict([adj_test, fea_test, score_test_])
```

```
251/251              0s 1ms/step
79/79              0s 949us/step
```

[32]:
```
ls_train_ = np.array(ls_train)
ls_test_ = np.array(ls_test)
```

[33]:
```
z_mean, _ = vae.encoder.predict([adj_test, fea_test, score_test_])
```

```
79/79                 0s 940us/step
```

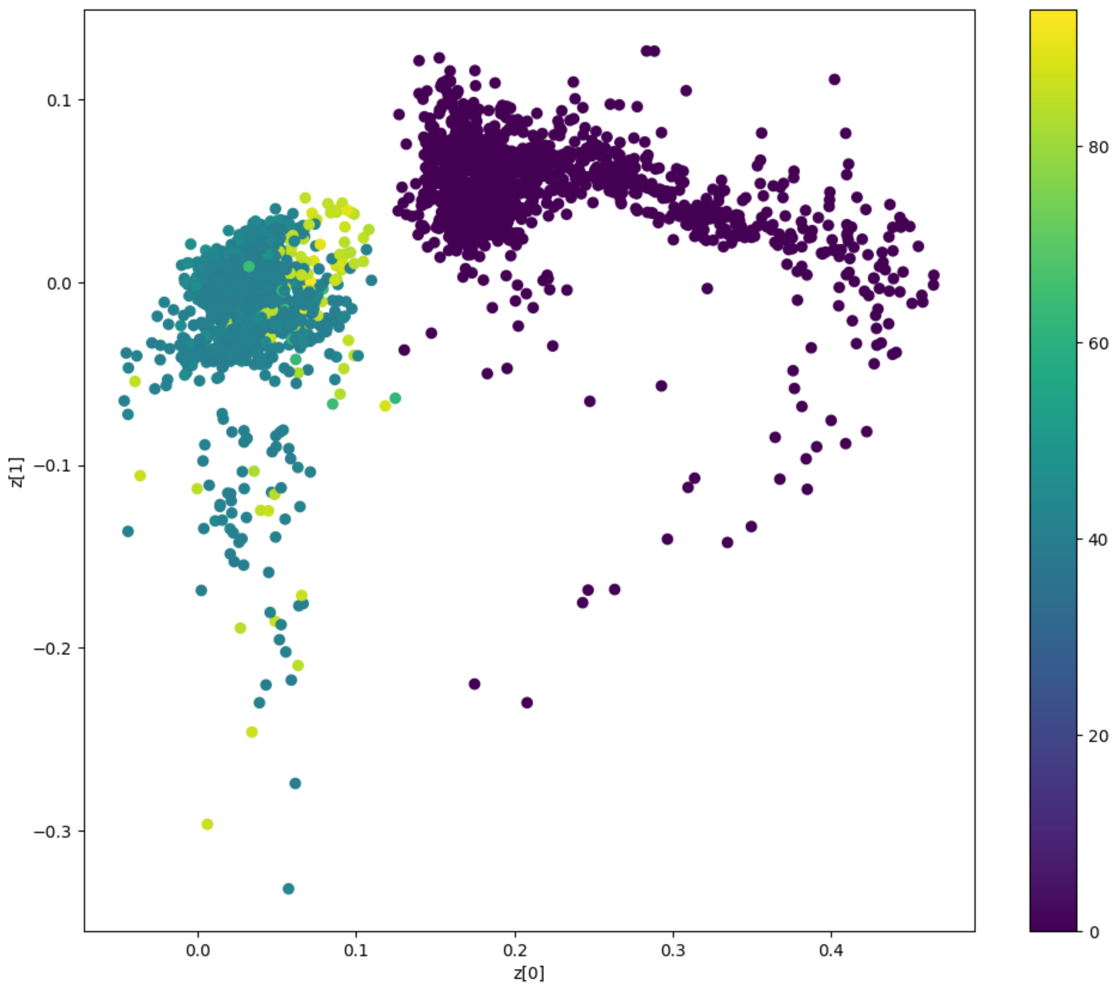
[34]:
```
latent_noise = np.random.normal(scale=0.1, size=z_mean.shape)  # Adjust scale␣
 ↪as needed
adj_pred, feature_pred = vae.decoder.predict([z_mean, score_test_])
print("Shape of adj_pred:", adj_pred.shape)
print("Shape of feature_pred:", feature_pred.shape)

# Reconstruct molecules
gen_molecules = [
    graph_to_molecule(adj_pred[i], feature_pred[i])
    for i in range(adj_pred.shape[0])
]
```

```
79/79                 0s 4ms/step
Shape of adj_pred: (2510, 5, 50, 50)
Shape of feature_pred: (2510, 50, 12)
```

[35]:
```
from scipy.stats import pearsonr

# Correlate latent dimensions with molecular scores
correlations = [pearsonr(z_mean[:, i], score_test_.flatten())[0] for i in␣
 ↪range(z_mean.shape[1])]
print("Correlations between latent dimensions and scores:", correlations)
```

```
Correlations between latent dimensions and scores: [-0.7850554335112523,
-0.6306591561123174, 0.7122755575526665, -0.8384445636545532,
-0.03329708578326372, -0.5128290639442652, -0.3503247027277769,
0.7418109621038158, 0.2802030373831089, 0.7067002149616265, -0.5584619795273105,
0.5889388012397376, -0.8340726274211244, -0.4677558525565457,
0.6963222732860327, -0.7603323000244127, 0.32131010068755494,
-0.025614220248301664, -0.35092335340120984, 0.21160725495285485,
-0.5861931198978443, -0.07654051722532812, 0.9091821075312202,
0.28357791502671437, 0.2875836981517368, -0.14522348188021028,
0.3581287029268357, 0.5954473220122992, 0.21280458240518516,
-0.24607783952217777, -0.3427265948240112, -0.866325531669493,
-0.0015810939225799243, -0.5049793164094454, -0.2222107454554603,
-0.03660636535373199, 0.8123520450428507, 0.7805129301284202,
-0.38020402282392707, -0.3575034298355539, -0.6015176190984987,
-0.2671712290984119, 0.7173269356104413, 0.7925537909553199,
-0.1711200109054484, -0.47689979486434786, -0.6228145335431732,
-0.10161235621550897, 0.26871469638137496, 0.4015489560252962,
0.43007015780231955, 0.8316529522907639, 0.28551271612804724,
0.5885021337372118, -0.6034185116847695, 0.7621254941182303, -0.479002061446721,
0.7086164492993798, 0.10362830692577474, -0.7154291969187991,
0.3289321510301183, 0.23354268279607698, -0.48967506805129474,
-0.11147307340246222, -0.4152879280556193, 0.2056593068335026,
0.1587648121037009, -0.8834500734164302, 0.29542531971612696,
0.4870743434230065, -0.12363232695242646, -0.013637741457221957,
-0.7754597517650521, -0.7327582691785419, -0.6256190122659977,
-0.714954677197715, 0.7890813489887689, -0.35633482994510884,
0.48801617925020113, 0.7360983673056387, -0.8001255948127852,
0.7061483860019719, 0.3824897644281933, 0.6199756543338328,
-0.22146019425794355, -0.03098114082251182, -0.5862217171707669,
-0.6391098003826348, 0.6026825853282776, -0.11006482487092431,
-0.7789412180675428, -0.46014937111018805, -0.30433131924323065,
-0.4758355433790832, 0.6251182532113478, -0.8814038613010716,
-0.4733470438418641, -0.5292510461568108, -0.74863584678701,
0.23352610233819618, 0.10242448041906899, -0.5309709488749564,
-0.7786801118734887, -0.6781878683546873, -0.15878406641304382,
-0.4466108024473777, 0.17192119083836072, 0.19463701849277398,
0.7019410208600592, -0.6527598279255493, 0.6438038557634551,
-0.7942390856552993, 0.7392068994549414, 0.4816113309242206, 0.5804107934560877,
-0.7167182671675619, 0.6972193487622782, 0.7071969203679852,
0.09713808146575374, 0.2176402763163122, -0.8495411644126821,
-0.4543197179974845, 0.30193883016333073, -0.8500648473261383,
-0.32764039233664466, 0.6982855275392338, 0.319589124112355,
-0.36319398549868454]
```

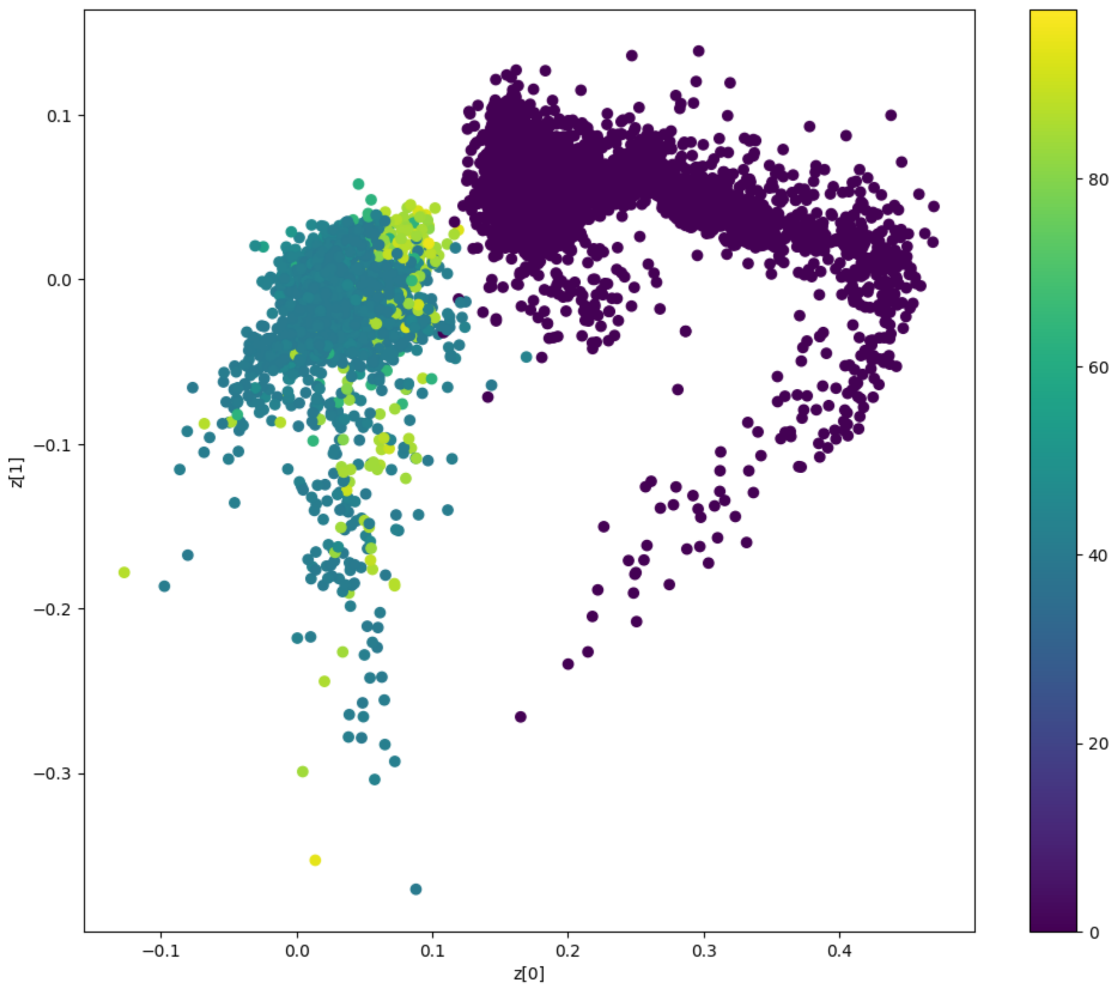
[36]:
```
plt.figure(figsize=(12, 10))
plt.scatter(z_mean[:, 0], z_mean[:, 1], c=score_test_)
plt.colorbar()
plt.xlabel("z[0]")
plt.ylabel("z[1]")
plt.show()
```



```
[37]: z_mean2, _2 = vae.encoder.predict([adj_train, fea_train, score_train_])
```

```
251/251                   0s 912us/step
```

```
[38]: plt.figure(figsize=(12, 10))
      plt.scatter(z_mean2[:, 0], z_mean2[:, 1], c=score_train_)
      plt.colorbar()
      plt.xlabel("z[0]")
      plt.ylabel("z[1]")
      plt.show()
```



```
[ ]:
```

0.12 Model Inferencing

We would be inferring our model to predict over random latent space and try to generate 100 new valid molecules.

0.12.1 Generate unique Molecules with the model

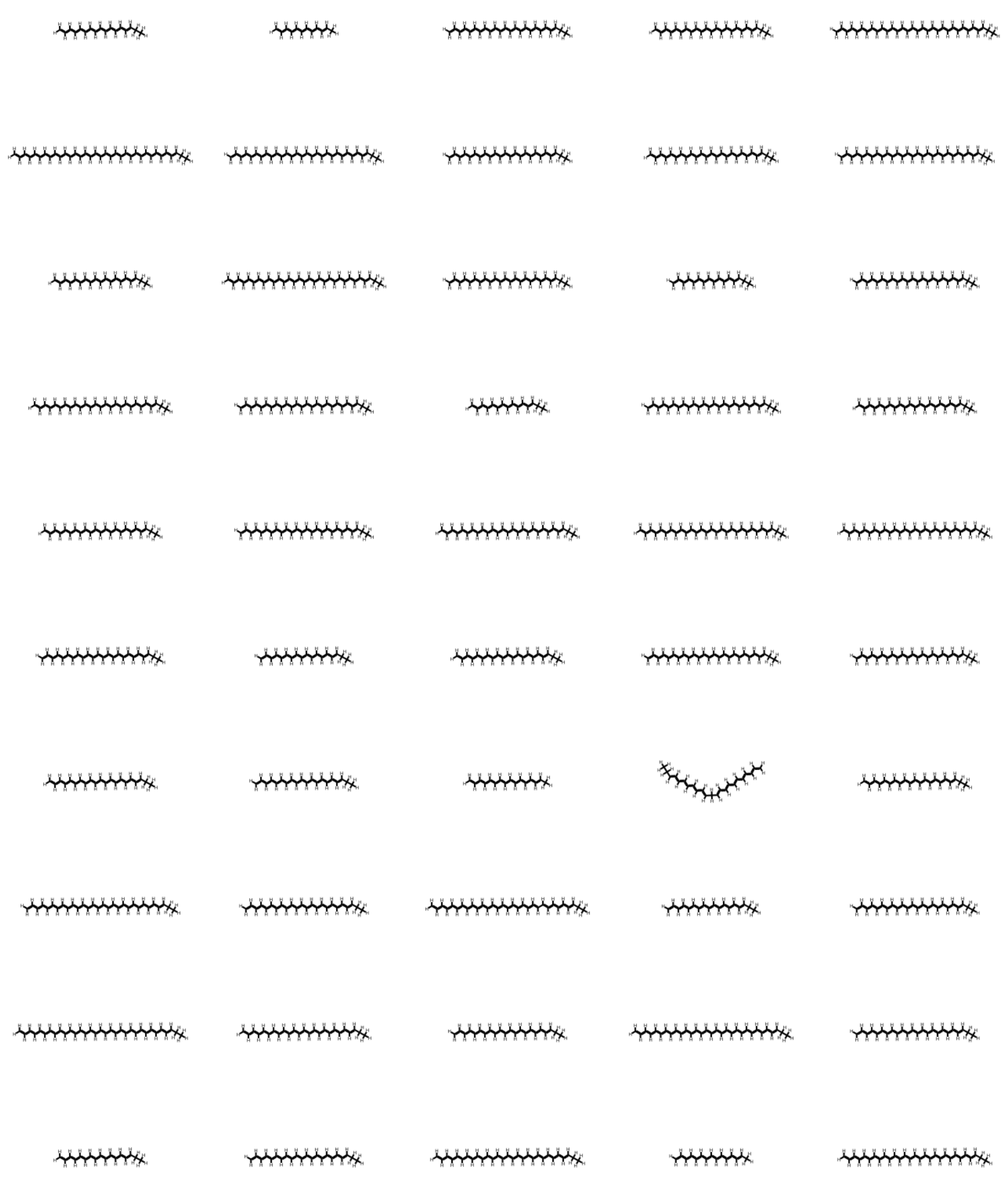
```
[39]: def inference(model=vae, batch_size=1000, dim = LATENT_DIM, activity=10):
          z = np.random.normal(size=(batch_size, dim))
          activityarray = (np.zeros(batch_size) + activity).reshape(-1,1)

          reconstruction_adjacency, reconstruction_features = model.decoder.
        ↪predict([z,activityarray])
          # obtain one-hot encoded adjacency tensor
```

```
    adjacency = tf.argmax(reconstruction_adjacency, axis=1)
    adjacency = tf.one_hot(adjacency, depth=BOND_DIM, axis=1)
    # Remove potential self-loops from adjacency
    adjacency = tf.linalg.set_diag(adjacency, tf.zeros(tf.shape(adjacency)[:
↪-1]))
    # obtain one-hot encoded feature tensor
    features = tf.argmax(reconstruction_features, axis=2)
    features = tf.one_hot(features, depth=ATOM_DIM, axis=2)

    return [
        graph_to_molecule(adjacency[i].numpy(), features[i].numpy())
        for i in range(batch_size)
    ]
```

[40]:
```
gen_mols = inference(batch_size=1000,activity=10)
MolsToGridImage([m for m in gen_mols if m is not None][:1000], molsPerRow=5,␣
↪subImgSize=(260, 160))
```

```
32/32                0s 4ms/step
Sanitization failed: Explicit valence for atom # 23 C, 8, is greater than
permitted
Sanitization failed: Explicit valence for atom # 2 N, 9, is greater than
permitted
Sanitization failed: Explicit valence for atom # 0 S, 101, is greater than
permitted

/Users/thinh/Library/Python/3.12/lib/python/site-
packages/rdkit/Chem/Draw/IPythonConsole.py:261: UserWarning: Truncating the list
of molecules to be displayed to 50. Change the maxMols value to display more.
  warnings.warn(
```

[40]:



[ ]: