final

December 12, 2024

0.1 Abstract

0.2 Package import

[81]:
```
import os

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

import ast
import numpy as np

from tensorflow import keras
```

[82]:
```
#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.*")
```

[83]:
```
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

[84]:
```
'''
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
```

[85]:
```
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]
```

[86]:
```
df_50
```

[86]:
```
                                                   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]
```

[87]:
```
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
```

[88]:
```
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)
```

[88]:

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

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

[89]:
```python
# 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
```

[90]:
```
filtered_df = balanced_df
filtered_df
```

[90]:

	SMILES	Activity	Score	Potency	Efficacy	Charged
0	CC1=C(C=CC=C1Br)NC(=O)C2=C(C=CS2)N3C=CC=C3	Active	82.0	8.9125	140.7280	False
1	CCCCCC(C(C)CC(=O)NC1CCCCC1)C(=O)O	Active	43.0	12.5893	136.6590	False
2	CC1=CC=C(C=C1)S(=O)(=O)NC2=NN3C(C=C(NC3=N2)C)C…	Active	41.0	22.3872	166.6580	False
3	CC1=CC(=O)OC2=C1C=C(C=C2)OCC(=O)NC3=CC=CC(=C3)…	Inactive	0.0	0.0000	0.0000	False
4	CC1=CC(=C(N1C)C)C(=O)COC(=O)C23CC4CC(C2)CC(C4)…	Inactive	0.0	0.0000	0.0000	False
…	…	…	…	…	…	…
12541	C1CN(CCN1C(=O)C2=CC=CC=C2CC3=CC=CC=C3)S(=O)(=O…	Inactive	0.0	0.0000	0.0000	False
12542	C1=CC=C(C=C1)OC2=NC=NC(=C2)N3C=NC=N3	Active	64.0	2.8184	74.9734	False
12543	CC1=C(C(=CC=C1)N2CCN(CC2)C3=NC4=CC=CC=C4C(=O)N…	Active	42.0	17.7828	126.5240	False
12544	CCC(C)NC(=O)CSC1=NC2=CC=CC=C2C3=NC(C(=O)N31)C4…	Active	42.0	15.8489	139.3040	False
12545	CC(C)C1=CC=C(C=C1)S(=O)(=O)NC2CCCC2	Inactive	0.0	0.0000	0.0000	False

[12546 rows x 6 columns]

0.4 Parameter setting

[91]:
```
'''
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']
```

[92]:
```
'''
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

[93]:
```
'''
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

[94]:
```
'''
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

[95]:
```
'''
    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

[96]:
```
'''
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

[97]:
```
'''
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

[98]:
```
'''
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)
```

[99]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

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

[100]:
```
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)
```

[101]:
```
print(np.max(score_train_n))
```

```
0.9999999999999999
```

[102]:
```
#Hyperparameters
BATCH_SIZE = 64
EPOCHS = 25
VAE_LR = 3e-4 # changed to 1e-3
LATENT_DIM = 64  # Size of the latent space
```

[103]:
```
'''
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 (InputLayer)	(None, 5, 50, 50)	0	-
feature_input (InputLayer)	(None, 50, 12)	0	-
relational_graph_c… (RelationalGraphCo…	(None, 50, 16)	960	adjacency_input[… feature_input[0]…
global_average_poo… (GlobalAveragePool…	(None, 16)	0	relational_graph…
score_input (InputLayer)	(None, 1)	0	-
concatenate_4 (Concatenate)	(None, 17)	0	global_average_p… score_input[0][0]
dense_14 (Dense)	(None, 256)	4,608	concatenate_4[0]…
dropout_10 (Dropout)	(None, 256)	0	dense_14[0][0]
dense_15 (Dense)	(None, 512)	131,584	dropout_10[0][0]
dropout_11 (Dropout)	(None, 512)	0	dense_15[0][0]
z_mean (Dense)	(None, 64)	32,832	dropout_11[0][0]
z_log_var (Dense)	(None, 64)	32,832	dropout_11[0][0]

Total params: 202,816 (792.25 KB)

Trainable params: 202,816 (792.25 KB)

Non-trainable params: 0 (0.00 B)

Model: "decoder"

Layer (type)	Output Shape	Param #	Connected to
latent_input (InputLayer)	(None, 64)	0	-
score_input (InputLayer)	(None, 1)	0	-
concatenate_5 (Concatenate)	(None, 65)	0	latent_input[0][… score_input[0][0]
dense_16 (Dense)	(None, 128)	8,448	concatenate_5[0]…
dropout_12 (Dropout)	(None, 128)	0	dense_16[0][0]
dense_17 (Dense)	(None, 256)	33,024	dropout_12[0][0]
dropout_13 (Dropout)	(None, 256)	0	dense_17[0][0]
dense_18 (Dense)	(None, 512)	131,584	dropout_13[0][0]
dropout_14 (Dropout)	(None, 512)	0	dense_18[0][0]
dense_19 (Dense)	(None, 12500)	6,412,500	dropout_14[0][0]
reshape_4 (Reshape)	(None, 5, 50, 50)	0	dense_19[0][0]
dense_20 (Dense)	(None, 600)	307,800	dropout_14[0][0]
symmetrize_layer_2 (SymmetrizeLayer)	(None, 5, 50, 50)	0	reshape_4[0][0]
reshape_5 (Reshape)	(None, 50, 12)	0	dense_20[0][0]
softmax_4 (Softmax)	(None, 5, 50, 50)	0	symmetrize_layer…

```
 softmax_5 (Softmax)   (None, 50, 12)              0   reshape_5[0][0]


Total params: 6,893,356 (26.30 MB)

Trainable params: 6,893,356 (26.30 MB)

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

[104]:
```
val_loss_list = []
train_loss_list = []
kl_theshold = 1.0
```

[105]:
```
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)
```

[106]:
```
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 17:40:47.192662: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 67.5119857788086, KL Loss: 6.928782939910889, Reconstruction Loss:
36.792850494384766

2024-12-12 17:40:47.744623: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 38.274810791015625, KL Loss: 9.519186973571777, Reconstruction
Loss: 34.92155838012695
BETA is:  0.05
Epoch 2/25

2024-12-12 17:40:55.860899: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 37.91454315185547, KL Loss: 7.56253719329834, Reconstruction Loss:
35.581417083740234

2024-12-12 17:40:56.367530: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 36.641273498535156, KL Loss: 8.278600692749023, Reconstruction
Loss: 34.01363754272461
BETA is:  0.05
Epoch 3/25

2024-12-12 17:41:04.417776: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 33.91452407836914, KL Loss: 12.668169021606445, Reconstruction Loss:
30.87645721435547

2024-12-12 17:41:04.926114: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 31.968669891357422, KL Loss: 14.902786254882812, Reconstruction
Loss: 28.769760131835938
BETA is:  0.05
Epoch 4/25
```

```
2024-12-12 17:41:12.943446: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 31.211721420288086, KL Loss: 14.64001178741455, Reconstruction Loss:
29.431264877319336

2024-12-12 17:41:13.447367: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.714691162109375, KL Loss: 16.28636360168457, Reconstruction
Loss: 27.554550170898438
BETA is:  0.05
Epoch 5/25

2024-12-12 17:41:21.540746: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 30.46052360534668, KL Loss: 15.52778434753418, Reconstruction Loss:
28.911928176879883

2024-12-12 17:41:22.047542: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 30.202251434326172, KL Loss: 17.129613876342773, Reconstruction
Loss: 26.783945083618164
BETA is:  0.05
Epoch 6/25

2024-12-12 17:41:30.180308: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.921600341796875, KL Loss: 15.809602737426758, Reconstruction
Loss: 28.653383255004883

2024-12-12 17:41:30.700175: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.859630584716797, KL Loss: 17.491769790649414, Reconstruction
Loss: 26.456256866455078
BETA is:  0.05
Epoch 7/25

2024-12-12 17:41:38.924941: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.628015518188477, KL Loss: 16.174915313720703, Reconstruction
Loss: 28.335111618041992

2024-12-12 17:41:39.429662: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.622241973876953, KL Loss: 17.435483932495117, Reconstruction
Loss: 26.7288875579834
BETA is:  0.05
Epoch 8/25
```

```
2024-12-12 17:41:47.334794: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.43010711669922, KL Loss: 15.276530265808105, Reconstruction Loss:
27.989168167114258

2024-12-12 17:41:47.837422: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.551408767700195, KL Loss: 16.294260025024414, Reconstruction
Loss: 26.6143798828125
BETA is:  0.05
Epoch 9/25

2024-12-12 17:41:55.819887: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.272197723388672, KL Loss: 15.636274337768555, Reconstruction
Loss: 27.817644119262695

2024-12-12 17:41:56.326923: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.30669403076172, KL Loss: 16.80072593688965, Reconstruction
Loss: 26.23880386352539
BETA is:  0.05
Epoch 10/25

2024-12-12 17:42:04.337097: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.058399200439453, KL Loss: 14.723729133605957, Reconstruction
Loss: 27.956562042236328

2024-12-12 17:42:04.853138: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.053916931152344, KL Loss: 15.159934997558594, Reconstruction
Loss: 26.115819931030273
BETA is:  0.05
Epoch 11/25

2024-12-12 17:42:12.986376: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.715190887451172, KL Loss: 11.514090538024902, Reconstruction
Loss: 28.518064498901367

2024-12-12 17:42:13.482674: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.690568923950195, KL Loss: 12.168988227844238, Reconstruction
Loss: 26.114303588867188
BETA is:  0.1
Epoch 12/25
```

```
2024-12-12 17:42:21.492328: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.64360237121582, KL Loss: 10.71100902557373, Reconstruction Loss:
28.185949325561523

2024-12-12 17:42:22.008832: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.729156494140625, KL Loss: 11.142773628234863, Reconstruction
Loss: 26.464569091796875
BETA is:  0.11
Epoch 13/25

2024-12-12 17:42:30.002394: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.666484832763672, KL Loss: 9.53032112121582, Reconstruction Loss:
28.10553741455078

2024-12-12 17:42:30.494278: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.79405403137207, KL Loss: 9.931119918823242, Reconstruction
Loss: 26.287099838256836
BETA is:  0.12
Epoch 14/25

2024-12-12 17:42:38.646808: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.585914611816406, KL Loss: 8.856047630310059, Reconstruction Loss:
28.223526000976562

2024-12-12 17:42:39.155758: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.59736442565918, KL Loss: 8.924586296081543, Reconstruction
Loss: 26.017650604248047
BETA is:  0.13
Epoch 15/25

2024-12-12 17:42:47.218385: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.609697341918945, KL Loss: 8.798980712890625, Reconstruction Loss:
27.70676612854004

2024-12-12 17:42:47.728926: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.519737243652344, KL Loss: 9.167922019958496, Reconstruction
Loss: 25.889205932617188
BETA is:  0.14
Epoch 16/25
```

```
2024-12-12 17:42:55.747121: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.487234115600586, KL Loss: 8.155357360839844, Reconstruction Loss:
28.30480194091797

2024-12-12 17:42:56.253404: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.83753204345703, KL Loss: 8.179804801940918, Reconstruction
Loss: 26.290420532226562
BETA is:  0.15
Epoch 17/25

2024-12-12 17:43:04.436814: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.83231544494629, KL Loss: 7.366892337799072, Reconstruction Loss:
27.722726821899414

2024-12-12 17:43:04.926401: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.76804542541504, KL Loss: 7.5256171226501465, Reconstruction
Loss: 25.996456146240234
BETA is:  0.16
Epoch 18/25

2024-12-12 17:43:13.155305: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.75334358215332, KL Loss: 7.2264509201049805, Reconstruction Loss:
27.751811981201172

2024-12-12 17:43:13.666420: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.657466888427734, KL Loss: 7.391184329986572, Reconstruction
Loss: 26.130847930908203
BETA is:  0.17
Epoch 19/25

2024-12-12 17:43:21.652274: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.453876495361328, KL Loss: 6.768893718719482, Reconstruction Loss:
27.775232315063477

2024-12-12 17:43:22.150796: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.46727752685547, KL Loss: 6.943620681762695, Reconstruction
Loss: 26.453027725219727
BETA is:  0.18
Epoch 20/25
```

2024-12-12 17:43:30.167847: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.316326141357422, KL Loss: 6.385615825653076, Reconstruction Loss: 27.53878402709961

2024-12-12 17:43:30.683688: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.50040626525879, KL Loss: 6.7170233726501465, Reconstruction Loss: 26.324064254760742
BETA is: 0.19
Epoch 21/25

2024-12-12 17:43:38.684038: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.39305305480957, KL Loss: 6.437716960906982, Reconstruction Loss: 27.668468475341797

2024-12-12 17:43:39.211773: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.559104919433594, KL Loss: 6.534706115722656, Reconstruction Loss: 26.013832092285156
BETA is: 0.2
Epoch 22/25

2024-12-12 17:43:47.245577: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.329505920410156, KL Loss: 6.323155879974365, Reconstruction Loss: 27.565624237060547

2024-12-12 17:43:47.723152: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.520280838012695, KL Loss: 6.5423903465271, Reconstruction Loss: 25.765941619873047
BETA is: 0.21
Epoch 23/25

2024-12-12 17:43:55.721353: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.349376678466797, KL Loss: 7.342165470123291, Reconstruction Loss: 27.528709411621094

2024-12-12 17:43:56.223708: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.69611930847168, KL Loss: 7.791652202606201, Reconstruction Loss: 25.623138427734375
BETA is: 0.22
Epoch 24/25

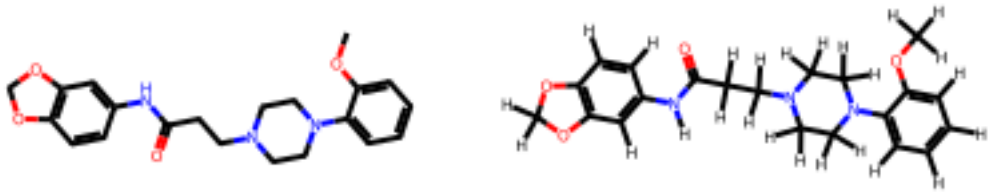
```
2024-12-12 17:44:04.237465: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.599098205566406, KL Loss: 6.041210651397705, Reconstruction Loss:
27.513463973999023

2024-12-12 17:44:04.752411: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Validation Loss: 29.530393600463867, KL Loss: 6.114850997924805, Reconstruction
Loss: 25.83179473876953
BETA is:  0.23
Epoch 25/25

2024-12-12 17:44:12.715896: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence

Train Loss: 29.29264259338379, KL Loss: 5.3024468421936035, Reconstruction Loss:
27.712268829345703
Validation Loss: 29.357452392578125, KL Loss: 5.430928707122803, Reconstruction
Loss: 25.732147216796875
BETA is:  0.24

2024-12-12 17:44:13.192882: W tensorflow/core/framework/local_rendezvous.cc:404]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

[107]:
```
'''
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)
```

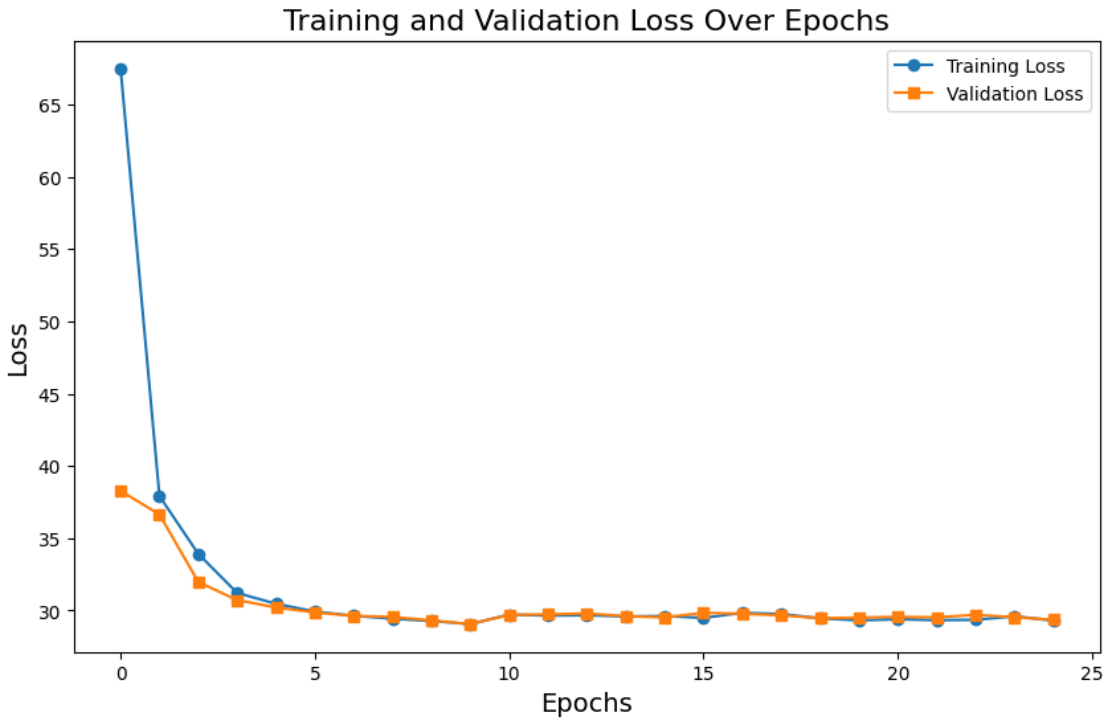
```
(1, 5, 50, 50)
(1, 50, 12)
(1, 1)
```

[108]:
```
mole_pred = graph_to_molecule(adj0[0], feature0[0])
Draw.MolsToGridImage([molobj,mole_pred], molsPerRow=2,)
```

[108]:



[109]:
```
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

[110]:
```
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_)
```

[111]:
```
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 816us/step
79/79             0s 752us/step
```

[112]:
```
ls_train_ = np.array(ls_train)
ls_test_ = np.array(ls_test)
```

[113]:
```
z_mean, _ = vae.encoder.predict([adj_test, fea_test, score_test_])
```

```
79/79             0s 784us/step
```

[114]:
```
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])
]
```

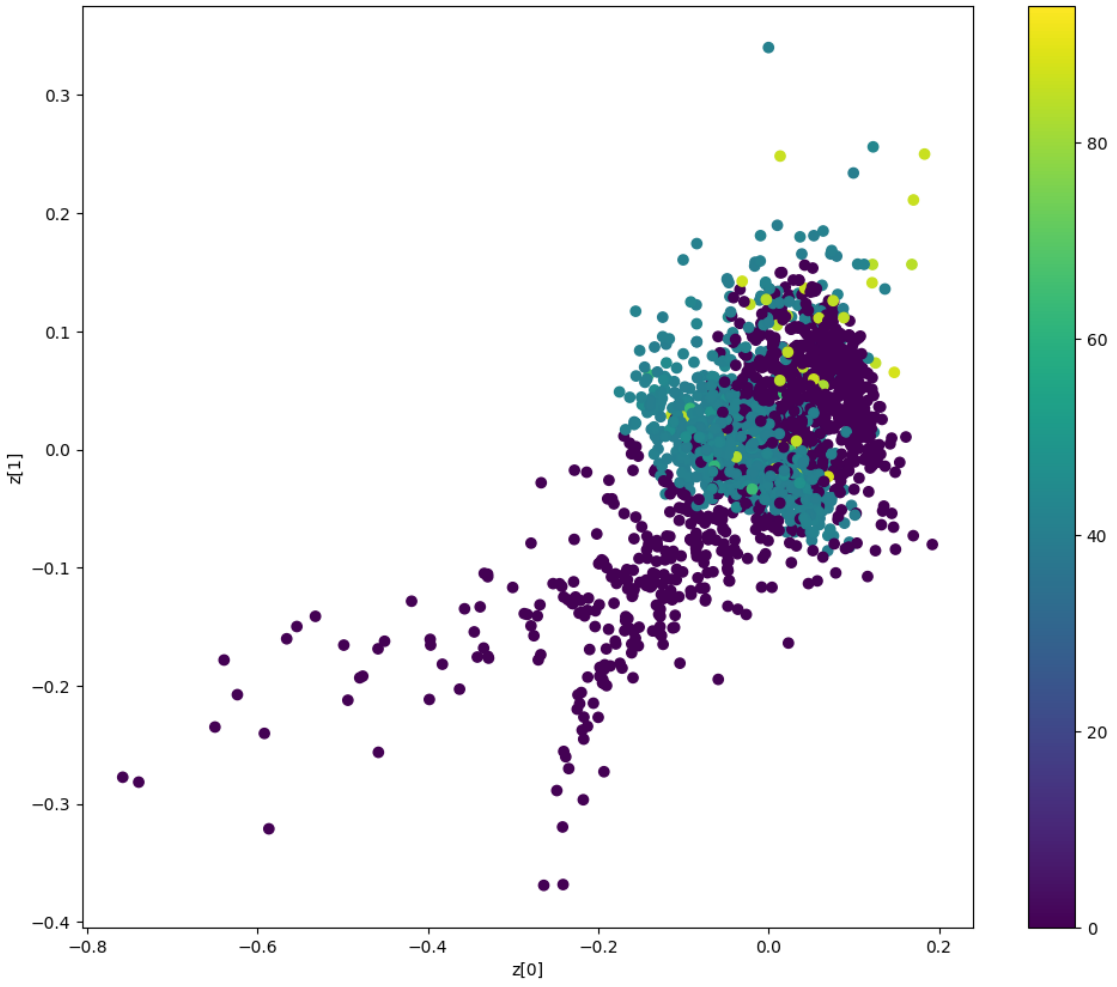
```
79/79             0s 4ms/step
Shape of adj_pred: (2510, 5, 50, 50)
Shape of feature_pred: (2510, 50, 12)
```

[115]:
```
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: [-0.004349401225860899,
0.0837755699388826, -0.25971438349604037, 0.01089905507168095,
0.2563236004379722, -0.46417415505428594, -0.19527687286425838,
-0.2550056207325086, -0.20254679655814944, -0.2331553594014547,
0.11238219537145551, -0.12127243999141679, -0.27250918831269566,
0.1437083015475163, 0.07772340646710407, -0.5119642594039573,
0.19574756283759895, -0.056254128352526966, 0.471479640295192,
0.2846070272700924, -0.1284014301161485, 0.32483209734609564,
0.2967585634203245, 0.4103324825148246, 0.15896315911821884,
-0.18116287617038773, -0.04274866804780583, -0.30124962433611585,
-0.8814724587345028, -0.320747991029133, -0.34324060666409084,
0.13402961539665476, -0.6858298025068398, 0.3586285148213044,
-0.14582722621482655, -0.15955799985761077, 0.06845070600322788,
0.16792003989719625, 0.14917820269286475, -0.440736801800347,
0.10924019093831303, 0.33527378645489425, 0.5019853114476431,
-0.2985487217912615, -0.8550841084691426, 0.0725337232811744,
0.5644988199279929, -0.4696211562016187, -0.06959425019528889,
-0.23640696953922574, -0.11170873639737328, 0.08060786112947425,
0.42289763954611853, 0.2279135019290195, 0.11821573509857296,
```

```
0.39064351958987137, -0.4218223397739742, -0.25221573691996324,
-0.3400466440900003, 0.6464342214679899, 0.35594304913160657,
-0.4595463919650813, 0.332005057237027, -0.18709303046119233]
```

[116]:
```
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()
```

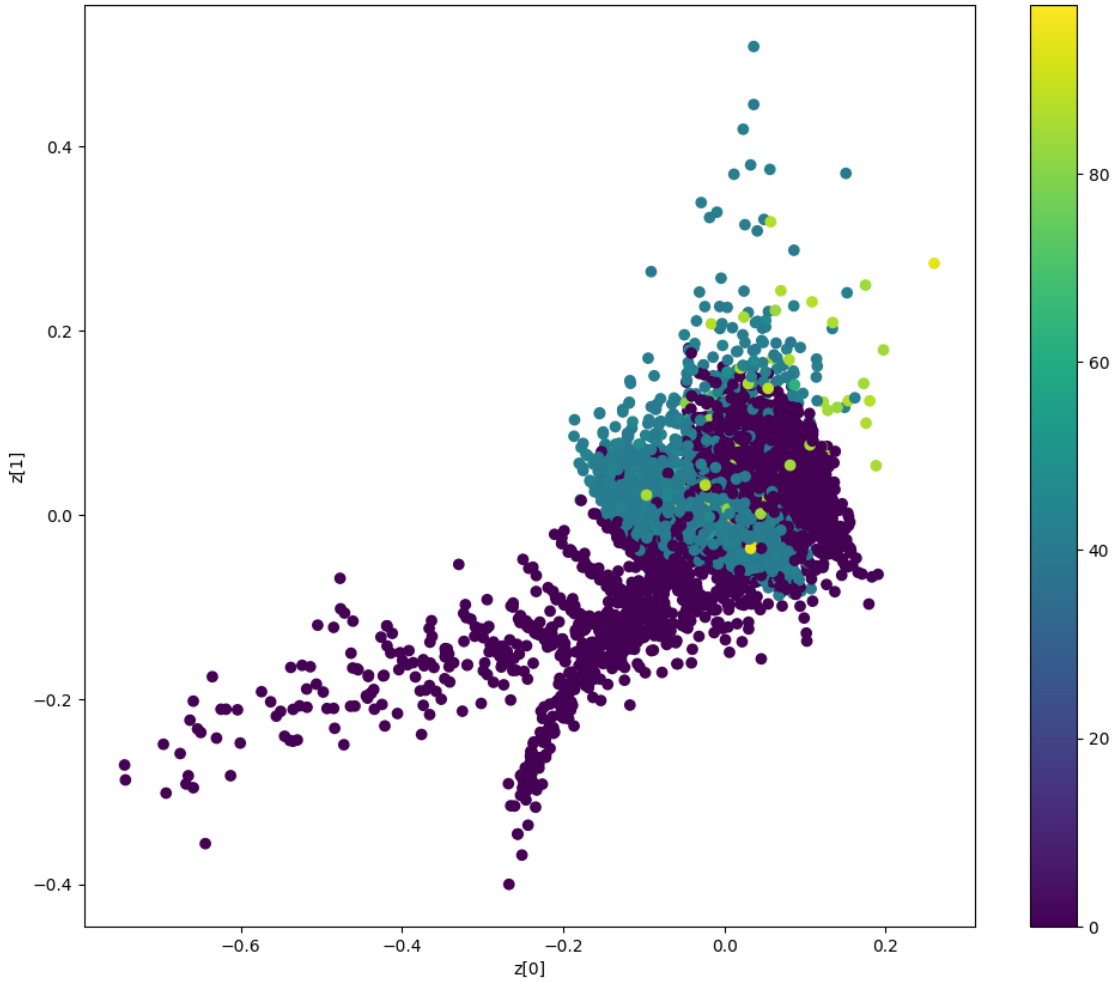


[117]:
```
z_mean2, _2 = vae.encoder.predict([adj_train, fea_train, score_train_])
```

```
251/251              0s 814us/step
```

[118]:
```
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.

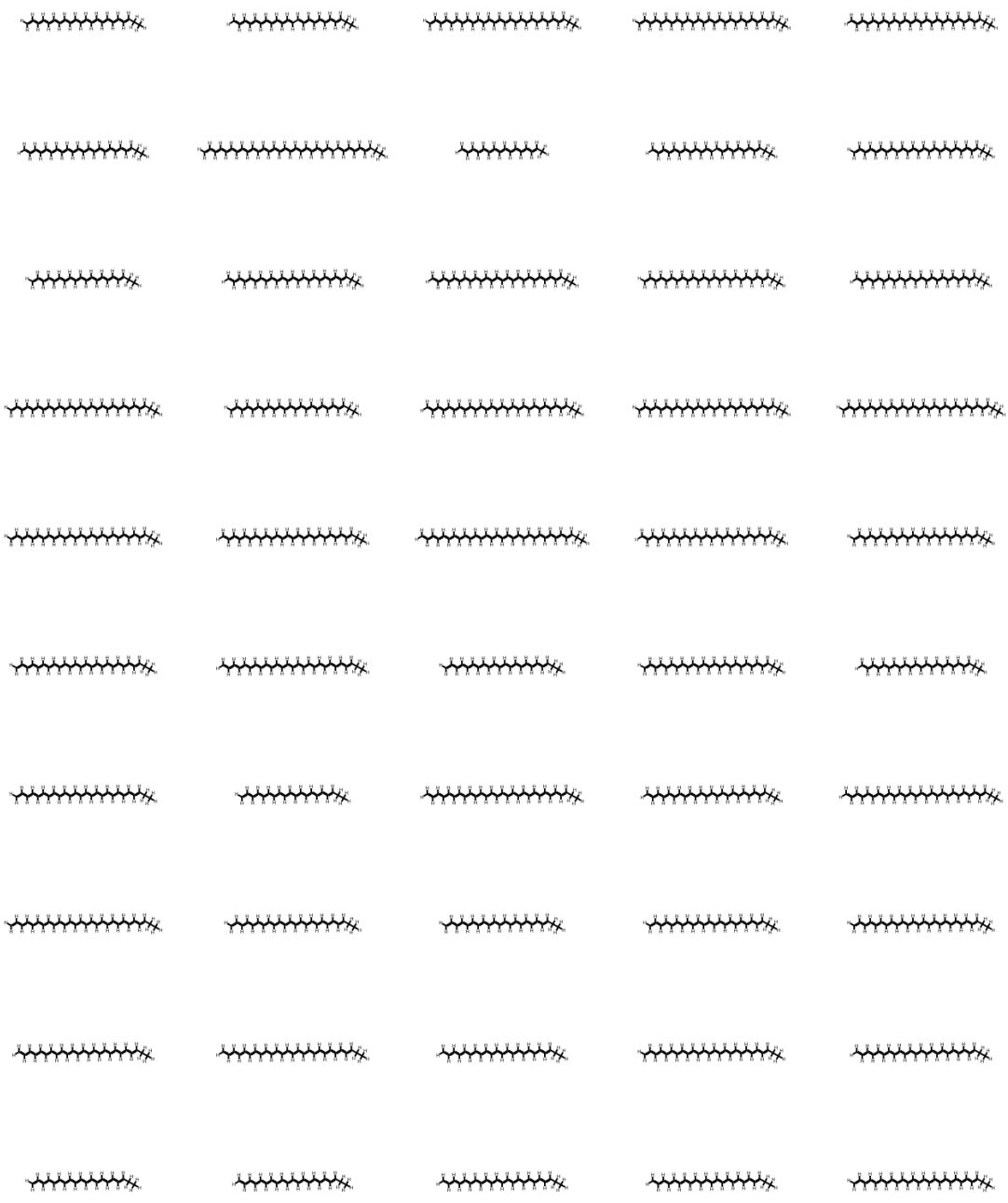
0.12.1 Generate unique Molecules with the model

[119]:
```
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)
    ]
```

[120]:
```
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 # 0 B, 30, 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(
```

[120]:



[ ]: