Installing rdkit for molecular visualization !

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

```
import numpy as np
import pandas as pd
import pprint
```

Changing dir to project dir

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

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

Reading 250k smiles dataset

```
df = pd.read_csv('250k_smiles.csv')
```

```
df.head()
```

	smiles	logP	qed	SAS
0	CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1\n	5.05060	0.702012	2.084095
1	C[C@@H]1CC(Nc2cncc(-c3nncn3C)c2)C[C@@H](C)C1\n	3.11370	0.928975	3.432004
2	N#Cc1ccc(-c2ccc(O[C@@H](C(=O)N3CCCC3)c3ccccc3)...	4.96778	0.599682	2.470633
3	CCOC(=O)[C@@H]1CCCN(C(=O)c2nc(-c3ccc(C)cc3)n3c...	4.00022	0.690944	2.822753
4	N#CC1=C(SCC(=O)Nc2cccc(Cl)c2)N=C([O-])[C@H](C#...	3.60956	0.789027	4.035182

```
df.smiles
```

```
0                 CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1\n
1            C[C@@H]1CC(Nc2cncc(-c3nncn3C)c2)C[C@@H](C)C1\n
2         N#Cc1ccc(-c2ccc(O[C@@H](C(=O)N3CCCC3)c3ccccc3)...
3         CCOC(=O)[C@@H]1CCCN(C(=O)c2nc(-c3ccc(C)cc3)n3c...
4         N#CC1=C(SCC(=O)Nc2cccc(Cl)c2)N=C([O-])[C@H](C#...
                              ...
249450    CC1(C)CC[C@H](CNC(=O)Cn2ncc3ccccc3c2=O)c2ccccc...
249451    Cn1ccnc1C(=O)c1ccc(NC(=O)C2CCN(C(=O)C(C)(C)C)C...
249452              Cc1ccc(NC(=O)C(=O)N(C)Cc2ccccc2)c(C)c1\n
249453         Cc1cc(C(=O)Nc2ccc(OCC(N)=O)cc2)c(C)n1C1CC1\n
249454    O=C(CC(c1ccccc1)c1ccccc1)N1CCN(S(=O)(=O)c2cccc...
Name: smiles, Length: 249455, dtype: object
```

```
df.smiles[0]
```

```
df.smiles[0]
```

```
'CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1\n'
```

```
from rdkit import Chem
```

```
m = Chem.MolFromSmiles(df.smiles[0])
m.GetNumAtoms()
```

```
24
```

Visualizing the molecules in Zinc Dataset

```
%matplotlib inline
import matplotlib.pyplot as plt
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit.Chem import AllChem
from rdkit.Chem import DataStructs
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import pandas as pd
import numpy as np
from scipy.spatial.distance import cdist
```

```
mols = [Chem.MolFromSmiles(smi) for smi in df.smiles]
sampleidx = np.random.choice(list(range(len(mols))), size=400, replace=False)
samplemols = [mols[i] for i in sampleidx]
```

```
sampleact = [9-np.log10(df['SAS'][idx]) for idx in sampleidx]
```

PCA t-SNE for getting chemical space and normalize the data.

```
fps = [AllChem.GetMorganFingerprintAsBitVect(m,2) for m in samplemols]
def fp2arr(fp):
    arr = np.zeros((0,))
    DataStructs.ConvertToNumpyArray(fp,arr)
    return arr
X = np.asarray([fp2arr(fp) for fp in fps])
```

```
X.shape
```

```
(400, 2048)
```

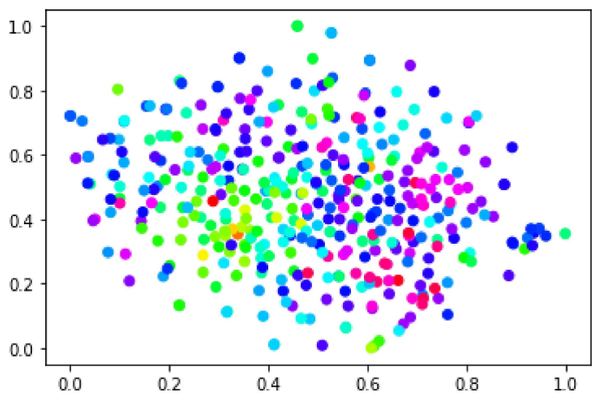
```
size = 20
N = size*size
data = PCA(n_components=100).fit_transform(X.astype(np.float32))
embeddings = TSNE(init='pca', random_state=794, verbose=2).fit_transform(data)
embeddings -= embeddings.min(axis=0)
embeddings /= embeddings.max(axis=0)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 400 samples in 0.008s...
[t-SNE] Computed neighbors for 400 samples in 0.102s...
[t-SNE] Computed conditional probabilities for sample 400 / 400
[t-SNE] Mean sigma: 1.794616
[t-SNE] Computed conditional probabilities in 0.032s
[t-SNE] Iteration 50: error = 79.5521317, gradient norm = 0.4340051 (50 iterations in 0.230s)
[t-SNE] Iteration 100: error = 83.9060211, gradient norm = 0.4077000 (50 iterations in 0.105s)
[t-SNE] Iteration 150: error = 80.9550018, gradient norm = 0.4482836 (50 iterations in 0.116s)
[t-SNE] Iteration 200: error = 80.7176361, gradient norm = 0.4596181 (50 iterations in 0.112s)
[t-SNE] Iteration 250: error = 80.9723587, gradient norm = 0.4496589 (50 iterations in 0.112s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 80.972359
[t-SNE] Iteration 300: error = 1.7843372, gradient norm = 0.0048324 (50 iterations in 0.102s)
[t-SNE] Iteration 350: error = 1.6662548, gradient norm = 0.0018741 (50 iterations in 0.084s)
[t-SNE] Iteration 400: error = 1.6069237, gradient norm = 0.0011403 (50 iterations in 0.087s)
[t-SNE] Iteration 450: error = 1.5744647, gradient norm = 0.0008867 (50 iterations in 0.088s)
[t-SNE] Iteration 500: error = 1.5500454, gradient norm = 0.0018959 (50 iterations in 0.084s)
[t-SNE] Iteration 550: error = 1.5304767, gradient norm = 0.0006488 (50 iterations in 0.086s)
[t-SNE] Iteration 600: error = 1.5107491, gradient norm = 0.0005210 (50 iterations in 0.088s)
[t-SNE] Iteration 650: error = 1.4964035, gradient norm = 0.0010575 (50 iterations in 0.084s)
[t-SNE] Iteration 700: error = 1.4850944, gradient norm = 0.0011142 (50 iterations in 0.099s)
[t-SNE] Iteration 750: error = 1.4780134, gradient norm = 0.0002108 (50 iterations in 0.089s)
[t-SNE] Iteration 800: error = 1.4720892, gradient norm = 0.0010717 (50 iterations in 0.082s)
[t-SNE] Iteration 850: error = 1.4640371, gradient norm = 0.0001661 (50 iterations in 0.088s)
[t-SNE] Iteration 900: error = 1.4633349, gradient norm = 0.0001187 (50 iterations in 0.086s)
[t-SNE] Iteration 950: error = 1.4631420, gradient norm = 0.0001028 (50 iterations in 0.082s)
[t-SNE] Iteration 1000: error = 1.4622216, gradient norm = 0.0001646 (50 iterations in 0.086s)
[t-SNE] KL divergence after 1000 iterations: 1.462222
```

Plotting the latent space of 400 molecules in the dataset

```
plt.scatter(embeddings[:,0], embeddings[:,1], c=sampleact, cmap='hsv')
```

```
<matplotlib.collections.PathCollection at 0x7f690d6b5518>
```

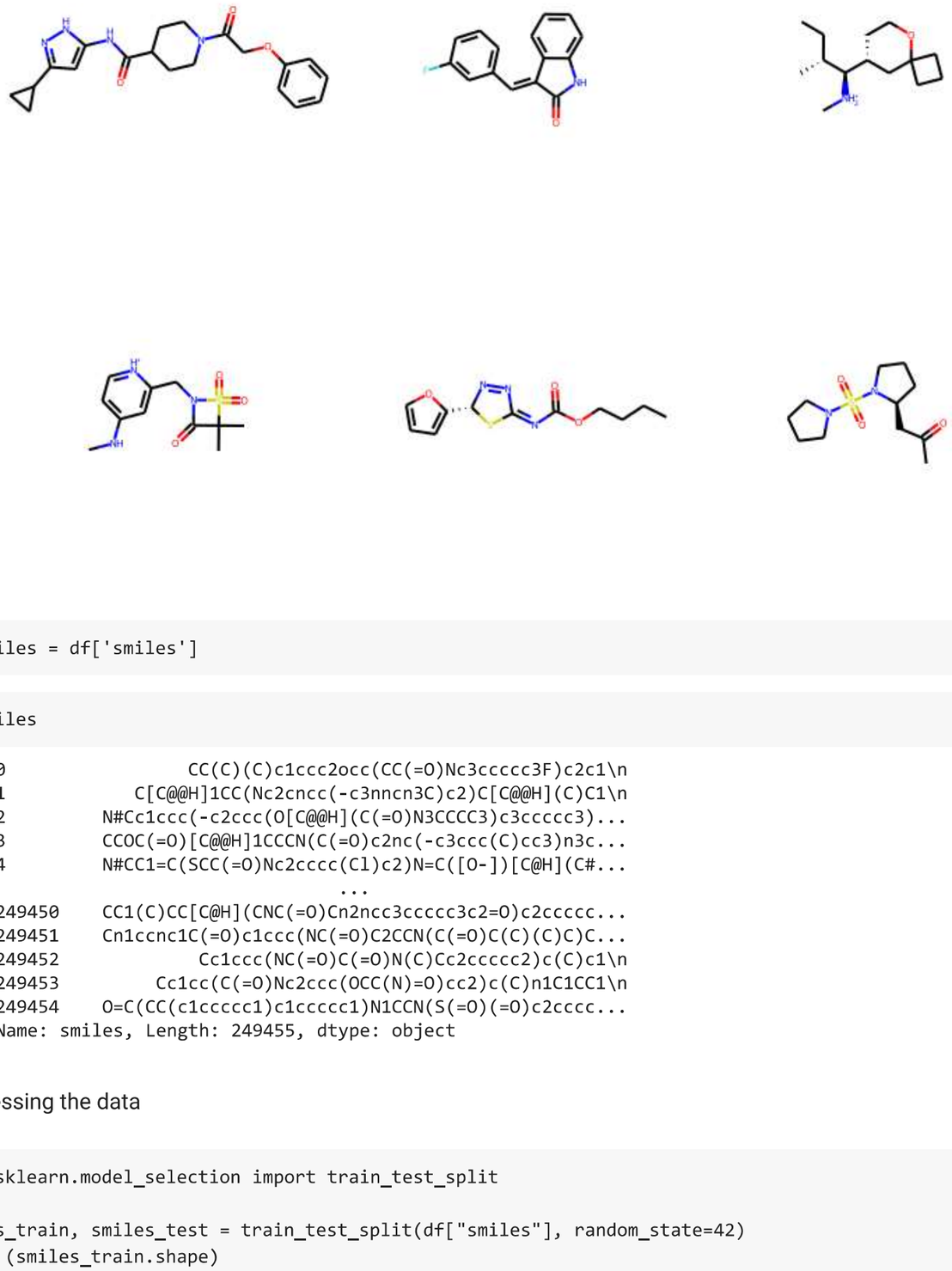



```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw
```

```
sampleidx = np.random.choice(list(range(len(mols))), size=6, replace=False)
samplemols = [mols[i] for i in sampleidx]
```

Plotting random molecules from dataset

```
Draw.MolsToGridImage(samplemols,molsPerRow=3,subImgSize=(250,250))
```



```
df_smiles = df['smiles']
```

```
df_smiles
```

```
0                 CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1\n
1            C[C@@H]1CC(Nc2cncc(-c3nncn3C)c2)C[C@@H](C)C1\n
2         N#Cc1ccc(-c2ccc(O[C@@H](C(=O)N3CCCC3)c3ccccc3)...
3         CCOC(=O)[C@@H]1CCCN(C(=O)c2nc(-c3ccc(C)cc3)n3c...
4         N#CC1=C(SCC(=O)Nc2cccc(Cl)c2)N=C([O-])[C@H](C#...
                                ...
249450    CC1(C)CC[C@H](CNC(=O)Cn2ncc3ccccc3c2=O)c2ccccc...
249451    Cn1ccnc1C(=O)c1ccc(NC(=O)C2CCN(C(=O)C(C)(C)C)C...
249452              Cc1ccc(NC(=O)C(=O)N(C)Cc2ccccc2)c(C)c1\n
249453         Cc1cc(C(=O)Nc2ccc(OCC(N)=O)cc2)c(C)n1C1CC1\n
249454    O=C(CC(c1ccccc1)c1ccccc1)N1CCN(S(=O)(=O)c2cccc...
Name: smiles, Length: 249455, dtype: object
```

Processing the data

```
from sklearn.model_selection import train_test_split

smiles_train, smiles_test = train_test_split(df["smiles"], random_state=42)
print (smiles_train.shape)
print (smiles_test.shape)
```

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

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

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

```
char_to_int
```

```
{'\n': 19,
 '!': 7,
 '#': 23,
 '(': 14,
 ')': 26,
 '+': 13,
 '-': 28,
 '/': 35,
 '1': 4,
 '2': 16,
 '3': 17,
 '4': 29,
 '5': 5,
 '6': 3,
 '7': 21,
 '8': 1,
 '=': 11,
 '@': 18,
 'B': 24,
 'C': 2,
 'E': 22,
 'F': 32,
 'H': 34,
 'I': 12,
 'N': 25,
 'O': 9,
 'P': 8,
 'S': 20,
 '[': 6,
 '\\': 36,
 ']': 10,
 'c': 31,
 'l': 33,
 'n': 15,
 'o': 30,
 'r': 0,
 's': 27}
```

```
len(charset)
```

```
37
```

```
X_train[8]
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
```

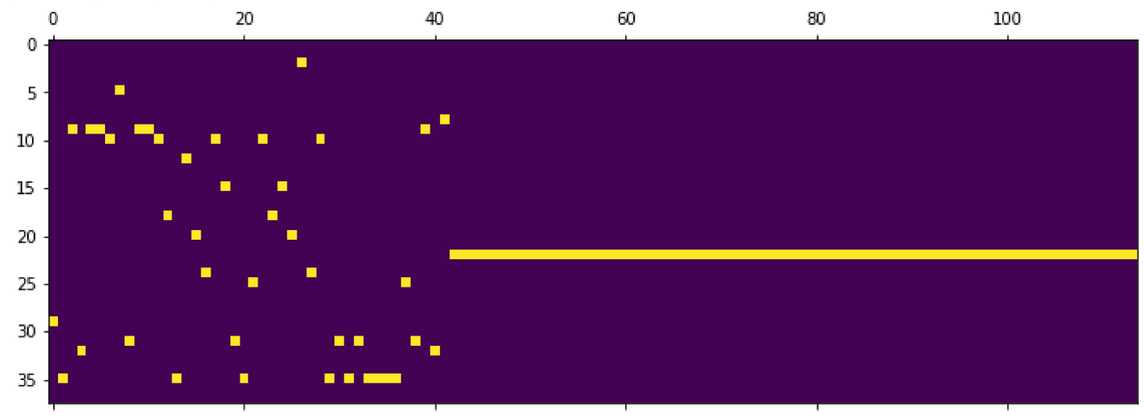
```
            ...,
            [0, 0, 0, ..., 0, 0, 0],
            [0, 0, 0, ..., 0, 0, 0],
            [0, 0, 0, ..., 0, 0, 0]], dtype=int8)
```

```
charset = [" ",'I', '\\', '+', 'P', 'B', 'F', '#', 'S', '\n', 'c', '(', '6', '@', '-', '/', 'N', '4', 'n', '[', '3', 'H', '7', 'E', 'l', ']', '2', 's', 'r', '8', '!', '5', ')', '1', '
print(len(charset))
```

```
38
```

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

```
Cc1cc(F)cc([C@H](N)C2([NH+](C)C)CCCC2)c1
(187091, 114, 38)
```



Creating defination functions for one hot encoding and decoding smiles

```
import h5py
import numpy as np
from rdkit import Chem


def one_hot_array(i, n):
    return list(map(int, [ix == i for ix in range(n)]))

def one_hot_index(vec, charset):
```

```
    return list(map(charset.index, vec))

def from_one_hot_array(vec):
    oh = np.where(vec == 1)
    if oh[0].shape == (0,):
        return None
    return int(oh[0][0])

def decode_smiles_from_indexes(vec, charset):
    return "".join(map(lambda x: charset[x], vec)).strip()
```

```
def encode_smiles(smiles, model, charset):
    cropped = list(smiles.ljust(114))
    preprocessed = np.array([list(map(lambda x: one_hot_array(x, len(charset)), one_hot_index(cropped, charset)))])
    latent = model.encoder.predict(preprocessed)
    return latent
```

```
def decode_latent_molecule(latent, model, charset, latent_dim):
    decoded = model.decoder.predict(latent.reshape(1, latent_dim)).argmax(axis=2)[0]
    smiles = decode_smiles_from_indexes(decoded, charset)
    return smiles
```

Creating interpolation function

```
def interpolate(source_smiles, dest_smiles, steps, charset, model, latent_dim):
    source_latent = encode_smiles(source_smiles, model, charset)
    dest_latent = encode_smiles(dest_smiles, model, charset)
    step = (dest_latent - source_latent) / float(steps)
    results = []
    for i in range(steps):
        item = source_latent + (step * i)
        decoded = decode_latent_molecule(item, model, charset, latent_dim)
        results.append(decoded)
    return results
```

```
def get_unique_mols(mol_list):
    inchi_keys = [Chem.InchiToInchiKey(Chem.MolToInchi(m)) for m in mol_list]
    u, indices = np.unique(inchi_keys, return_index=True)
    unique_mols = [[mol_list[i], inchi_keys[i]] for i in indices]
    return unique_mols
```

Building the model architecture for VAE

Reference : https://arxiv.org/pdf/1610.02415.pdf

```
import copy
from keras import backend as K
from keras import objectives
from keras.models import Model
from keras.layers import Input, Dense, Lambda
from keras.layers.core import Dense, Activation, Flatten, RepeatVector
from keras.layers.wrappers import TimeDistributed
from keras.layers.recurrent import GRU
from keras.layers.convolutional import Convolution1D
```

```
class MoleculeVAE():

    autoencoder = None

    def create(self,
               charset,
               max_length = 114,
               latent_rep_size = 114,
               weights_file = None):
        charset_length = len(charset)

        x = Input(shape=(max_length, charset_length))
        _, z = self._buildEncoder(x, latent_rep_size, max_length)
        self.encoder = Model(x, z)

        encoded_input = Input(shape=(latent_rep_size,))
        self.decoder = Model(
            encoded_input,
            self._buildDecoder(
                encoded_input,
                latent_rep_size,
                max_length,
                charset_length
            )
        )

        x1 = Input(shape=(max_length, charset_length))
        vae_loss, z1 = self._buildEncoder(x1, latent_rep_size, max_length)
        self.autoencoder = Model(
            x1,
            self._buildDecoder(
                z1,
                latent_rep_size,
                max_length,
                charset_length
            )
        )

        if weights_file:
            self.autoencoder.load_weights(weights_file)
            self.encoder.load_weights(weights_file, by_name = True)
            self.decoder.load_weights(weights_file, by_name = True)

        self.autoencoder.compile(optimizer = 'Adam',
                                 loss = vae_loss,
                                 metrics = ['accuracy'])

    def _buildEncoder(self, x, latent_rep_size, max_length, epsilon_std = 0.01):
        h = Convolution1D(9, 9, activation = 'relu', name='conv_1')(x)
        h = Convolution1D(9, 9, activation = 'relu', name='conv_2')(h)
        h = Convolution1D(10, 11, activation = 'relu', name='conv_3')(h)
        h = Flatten(name='flatten_1')(h)
        h = Dense(435, activation = 'relu', name='dense_1')(h)

        def sampling(args):
            z_mean_, z_log_var_ = args
            batch_size = K.shape(z_mean_)[0]
            epsilon = K.random_normal(shape=(batch_size, latent_rep_size), mean=0., stddev = epsilon_std)
            return z_mean_ + K.exp(z_log_var_ / 2) * epsilon
```

```
        return z_mean_ + K.exp(z_log_var_ / 2) * epsilon

    z_mean = Dense(latent_rep_size, name='z_mean', activation = 'linear')(h)
    z_log_var = Dense(latent_rep_size, name='z_log_var', activation = 'linear')(h)

    def vae_loss(x, x_decoded_mean):
        x = K.flatten(x)
        x_decoded_mean = K.flatten(x_decoded_mean)
        xent_loss = max_length * objectives.binary_crossentropy(x, x_decoded_mean)
        kl_loss = - 0.5 * K.mean(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis = -1)
        return xent_loss + kl_loss

    return (vae_loss, Lambda(sampling, output_shape=(latent_rep_size,), name='lambda')([z_mean, z_log_var]))

def _buildDecoder(self, z, latent_rep_size, max_length, charset_length):
    h = Dense(latent_rep_size, name='latent_input', activation = 'relu')(z)
    h = RepeatVector(max_length, name='repeat_vector')(h)
    h = GRU(501, return_sequences = True, name='gru_1')(h)
    h = GRU(501, return_sequences = True, name='gru_2')(h)
    h = GRU(501, return_sequences = True, name='gru_3')(h)
    return TimeDistributed(Dense(charset_length, activation='softmax'), name='decoded_mean')(h)

def save(self, filename):
    self.autoencoder.save_weights(filename)

def load(self, charset, weights_file, latent_rep_size = 114):
    self.create(charset, weights_file = weights_file, latent_rep_size = latent_rep_size)
```

Importing keras backend

```
import tensorflow.compat.v1.keras.backend as K
import tensorflow as tf
tf.compat.v1.disable_eager_execution()
```

```
latent_dim = 114
```

```
model = MoleculeVAE()
```

```
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
```

```
checkpointer = ModelCheckpoint(filepath = '/content/drive/MyDrive/ML_2/project/best_model.h5',
                               verbose = 1,
                               save_best_only = True)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              min_lr = 0.0001)
```

Creating the model and training !

```
model.create(charset, latent_rep_size)

checkpointer = ModelCheckpoint(filepath = '/content/drive/MyDrive/ML_2/project/weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5',
```

```
                                verbose = 1,
                                save_best_only = True)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              min_lr = 0.0001)

model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 5,
        batch_size = 600,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
Train on 187091 samples, validate on 62364 samples
Epoch 1/5
187091/187091 [==============================] - ETA: 0s - loss: 3.1805 - accuracy: 0.6700/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.
  warnings.warn('`Model.state_updates` will be removed in a future version. '

Epoch 00001: val_loss improved from inf to 2.69514, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 134s 718us/sample - loss: 3.1805 - accuracy: 0.6700 - val_loss: 2.6951 - val_accuracy: 0.6934
Epoch 2/5
187091/187091 [==============================] - ETA: 0s - loss: 2.7302 - accuracy: 0.6900
Epoch 00002: val_loss improved from 2.69514 to 2.67614, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 130s 697us/sample - loss: 2.7302 - accuracy: 0.6900 - val_loss: 2.6761 - val_accuracy: 0.6911
Epoch 3/5
187091/187091 [==============================] - ETA: 0s - loss: 2.6652 - accuracy: 0.6984
Epoch 00003: val_loss improved from 2.67614 to 2.60478, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 128s 685us/sample - loss: 2.6652 - accuracy: 0.6984 - val_loss: 2.6048 - val_accuracy: 0.7073
Epoch 4/5
187091/187091 [==============================] - ETA: 0s - loss: 2.6049 - accuracy: 0.7115
Epoch 00004: val_loss improved from 2.60478 to 2.57261, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 129s 689us/sample - loss: 2.6049 - accuracy: 0.7115 - val_loss: 2.5726 - val_accuracy: 0.7121
Epoch 5/5
187091/187091 [==============================] - ETA: 0s - loss: 2.5210 - accuracy: 0.7233
Epoch 00005: val_loss improved from 2.57261 to 2.47377, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 128s 684us/sample - loss: 2.5210 - accuracy: 0.7233 - val_loss: 2.4738 - val_accuracy: 0.7262
<tensorflow.python.keras.callbacks.History at 0x7f64153c0ef0>
```

Training the model in batches not to overload memory

```
model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 15,
```

```
    batch_size = 1200,
   callbacks = [checkpointer, reduce_lr],
   validation_data = (X_test,Y_test))
```

```
187091/187091 [==============================] - 95s 508us/sample - loss: 2.3028 - accuracy: 0.7463 - val_loss: 2.3479 - val_accuracy: 0.7408
Epoch 4/15
187091/187091 [==============================] - ETA: 0s - loss: 2.2564 - accuracy: 0.7505
Epoch 00004: val_loss improved from 2.30983 to 2.19369, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 545us/sample - loss: 2.2564 - accuracy: 0.7505 - val_loss: 2.1937 - val_accuracy: 0.7600
Epoch 5/15
187091/187091 [==============================] - ETA: 0s - loss: 2.1737 - accuracy: 0.7569
Epoch 00005: val_loss improved from 2.19369 to 2.15747, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 546us/sample - loss: 2.1737 - accuracy: 0.7569 - val_loss: 2.1575 - val_accuracy: 0.7606
Epoch 6/15
187091/187091 [==============================] - ETA: 0s - loss: 2.1035 - accuracy: 0.7646
Epoch 00006: val_loss improved from 2.15747 to 2.06249, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 548us/sample - loss: 2.1035 - accuracy: 0.7646 - val_loss: 2.0625 - val_accuracy: 0.7697
Epoch 7/15
187091/187091 [==============================] - ETA: 0s - loss: 2.0112 - accuracy: 0.7758
Epoch 00007: val_loss improved from 2.06249 to 1.93333, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 547us/sample - loss: 2.0112 - accuracy: 0.7758 - val_loss: 1.9333 - val_accuracy: 0.7881
Epoch 8/15
187091/187091 [==============================] - ETA: 0s - loss: 1.9476 - accuracy: 0.7836
Epoch 00008: val_loss did not improve from 1.93333
187091/187091 [==============================] - 95s 508us/sample - loss: 1.9476 - accuracy: 0.7836 - val_loss: 1.9435 - val_accuracy: 0.7799
Epoch 9/15
187091/187091 [==============================] - ETA: 0s - loss: 1.8681 - accuracy: 0.7929
Epoch 00009: val_loss improved from 1.93333 to 1.80764, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 103s 548us/sample - loss: 1.8681 - accuracy: 0.7929 - val_loss: 1.8076 - val_accuracy: 0.8017
Epoch 10/15
187091/187091 [==============================] - ETA: 0s - loss: 1.8248 - accuracy: 0.7984
Epoch 00010: val_loss improved from 1.80764 to 1.74569, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 547us/sample - loss: 1.8248 - accuracy: 0.7984 - val_loss: 1.7457 - val_accuracy: 0.8114
Epoch 11/15
187091/187091 [==============================] - ETA: 0s - loss: 1.7536 - accuracy: 0.8080
Epoch 00011: val_loss improved from 1.74569 to 1.66567, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 547us/sample - loss: 1.7536 - accuracy: 0.8080 - val_loss: 1.6657 - val_accuracy: 0.8198
Epoch 12/15
187091/187091 [==============================] - ETA: 0s - loss: 1.6956 - accuracy: 0.8154
Epoch 00012: val_loss did not improve from 1.66567
187091/187091 [==============================] - 95s 508us/sample - loss: 1.6956 - accuracy: 0.8154 - val_loss: 1.9286 - val_accuracy: 0.7846
Epoch 13/15
187091/187091 [==============================] - ETA: 0s - loss: 1.6595 - accuracy: 0.8206
Epoch 00013: val_loss improved from 1.66567 to 1.59636, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 547us/sample - loss: 1.6595 - accuracy: 0.8206 - val_loss: 1.5964 - val_accuracy: 0.8264
Epoch 14/15
187091/187091 [==============================] - ETA: 0s - loss: 1.5785 - accuracy: 0.8309
Epoch 00014: val_loss did not improve from 1.59636
187091/187091 [==============================] - 95s 508us/sample - loss: 1.5785 - accuracy: 0.8309 - val_loss: 1.6976 - val_accuracy: 0.8170
Epoch 15/15
187091/187091 [==============================] - ETA: 0s - loss: 1.5657 - accuracy: 0.8334
Epoch 00015: val_loss improved from 1.59636 to 1.51263, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 544us/sample - loss: 1.5657 - accuracy: 0.8334 - val_loss: 1.5126 - val_accuracy: 0.8411
<tensorflow.python.keras.callbacks.History at 0x7f6413f7dd30>
```

```
model.autoencoder.fit(
```

```
model.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 15,
        batch_size = 1200,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
187091/187091 [==============================] - 102s 546us/sample - loss: 1.3899 - accuracy: 0.8551 - val_loss: 1.2722 - val_accuracy: 0.8686
Epoch 4/15
187091/187091 [==============================] - ETA: 0s - loss: 1.3007 - accuracy: 0.8656
Epoch 00004: val_loss improved from 1.27225 to 1.21755, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 544us/sample - loss: 1.3007 - accuracy: 0.8656 - val_loss: 1.2175 - val_accuracy: 0.8759
Epoch 5/15
187091/187091 [==============================] - ETA: 0s - loss: 1.2385 - accuracy: 0.8730
Epoch 00005: val_loss improved from 1.21755 to 1.17656, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 545us/sample - loss: 1.2385 - accuracy: 0.8730 - val_loss: 1.1766 - val_accuracy: 0.8797
Epoch 6/15
187091/187091 [==============================] - ETA: 0s - loss: 1.2804 - accuracy: 0.8685
Epoch 00006: val_loss improved from 1.17656 to 1.15503, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 101s 541us/sample - loss: 1.2804 - accuracy: 0.8685 - val_loss: 1.1550 - val_accuracy: 0.8824
Epoch 7/15
187091/187091 [==============================] - ETA: 0s - loss: 1.1875 - accuracy: 0.8793
Epoch 00007: val_loss improved from 1.15503 to 1.08254, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 544us/sample - loss: 1.1875 - accuracy: 0.8793 - val_loss: 1.0825 - val_accuracy: 0.8905
Epoch 8/15
187091/187091 [==============================] - ETA: 0s - loss: 1.1611 - accuracy: 0.8820
Epoch 00008: val_loss did not improve from 1.08254
187091/187091 [==============================] - 95s 507us/sample - loss: 1.1611 - accuracy: 0.8820 - val_loss: 1.2486 - val_accuracy: 0.8693
Epoch 9/15
187091/187091 [==============================] - ETA: 0s - loss: 1.1038 - accuracy: 0.8888
Epoch 00009: val_loss improved from 1.08254 to 1.00526, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 545us/sample - loss: 1.1038 - accuracy: 0.8888 - val_loss: 1.0053 - val_accuracy: 0.8996
Epoch 10/15
187091/187091 [==============================] - ETA: 0s - loss: 1.1120 - accuracy: 0.8882
Epoch 00010: val_loss did not improve from 1.00526
187091/187091 [==============================] - 95s 507us/sample - loss: 1.1120 - accuracy: 0.8882 - val_loss: 1.1260 - val_accuracy: 0.8855
Epoch 11/15
187091/187091 [==============================] - ETA: 0s - loss: 1.0751 - accuracy: 0.8918
Epoch 00011: val_loss improved from 1.00526 to 0.99732, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 547us/sample - loss: 1.0751 - accuracy: 0.8918 - val_loss: 0.9973 - val_accuracy: 0.9001
Epoch 12/15
187091/187091 [==============================] - ETA: 0s - loss: 0.9941 - accuracy: 0.9008

Epoch 00012: val_loss improved from 0.99732 to 0.90793, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 546us/sample - loss: 0.9941 - accuracy: 0.9008 - val_loss: 0.9079 - val_accuracy: 0.9096
Epoch 13/15
187091/187091 [==============================] - ETA: 0s - loss: 1.0024 - accuracy: 0.8999
Epoch 00013: val_loss did not improve from 0.90793
187091/187091 [==============================] - 95s 507us/sample - loss: 1.0024 - accuracy: 0.8999 - val_loss: 1.0191 - val_accuracy: 0.8964
Epoch 14/15
187091/187091 [==============================] - ETA: 0s - loss: 0.9457 - accuracy: 0.9058
Epoch 00014: val_loss improved from 0.90793 to 0.88971, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
187091/187091 [==============================] - 102s 546us/sample - loss: 0.9457 - accuracy: 0.9058 - val_loss: 0.8897 - val_accuracy: 0.9116
Epoch 15/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8946 - accuracy: 0.9114
Epoch 00015: val_loss improved from 0.88971 to 0.82152, saving model to /content/drive/MyDrive/ML_2/project
INFO:tensorflow:Assets written to: /content/drive/MyDrive/ML_2/project/assets
```

```
187091/187091 [==============================] - 101s 541us/sample - loss: 0.8946 - accuracy: 0.9114 - val_loss: 0.8215 - val_accuracy: 0.9190
<tensorflow.python.keras.callbacks.History at 0x7f690d61f278>
```

```
model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 15,
        batch_size = 1200,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
Epoch 00001: val_loss improved from inf to 0.85052, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-01-0.92.hdf5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.9954 - accuracy: 0.9012 - val_loss: 0.8505 - val_accuracy: 0.9164
Epoch 2/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8658 - accuracy: 0.9146
Epoch 00002: val_loss improved from 0.85052 to 0.78736, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-02-0.92.hdf5
187091/187091 [==============================] - 95s 506us/sample - loss: 0.8658 - accuracy: 0.9146 - val_loss: 0.7874 - val_accuracy: 0.9226
Epoch 3/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8468 - accuracy: 0.9165
Epoch 00003: val_loss improved from 0.78736 to 0.76765, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-03-0.92.hdf5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.8468 - accuracy: 0.9165 - val_loss: 0.7676 - val_accuracy: 0.9241
Epoch 4/15
187091/187091 [==============================] - ETA: 0s - loss: 0.9041 - accuracy: 0.9105
Epoch 00004: val_loss did not improve from 0.76765
187091/187091 [==============================] - 95s 507us/sample - loss: 0.9041 - accuracy: 0.9105 - val_loss: 0.7817 - val_accuracy: 0.9233
Epoch 5/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7388 - accuracy: 0.9274
Epoch 00005: val_loss improved from 0.76765 to 0.73671, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-05-0.93.hdf5
187091/187091 [==============================] - 95s 507us/sample - loss: 0.7388 - accuracy: 0.9274 - val_loss: 0.7367 - val_accuracy: 0.9275
Epoch 6/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8396 - accuracy: 0.9179
Epoch 00006: val_loss improved from 0.73671 to 0.70474, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-06-0.93.hdf5
187091/187091 [==============================] - 95s 510us/sample - loss: 0.8396 - accuracy: 0.9179 - val_loss: 0.7047 - val_accuracy: 0.9309
Epoch 7/15
187091/187091 [==============================] - ETA: 0s - loss: 0.9386 - accuracy: 0.9080
Epoch 00007: val_loss did not improve from 0.70474
187091/187091 [==============================] - 95s 507us/sample - loss: 0.9386 - accuracy: 0.9080 - val_loss: 0.8940 - val_accuracy: 0.9107
Epoch 8/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8040 - accuracy: 0.9212
Epoch 00008: val_loss improved from 0.70474 to 0.68513, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-08-0.93.hdf5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.8040 - accuracy: 0.9212 - val_loss: 0.6851 - val_accuracy: 0.9331
Epoch 9/15
187091/187091 [==============================] - ETA: 0s - loss: 0.8446 - accuracy: 0.9174
Epoch 00009: val_loss did not improve from 0.68513
187091/187091 [==============================] - 95s 508us/sample - loss: 0.8446 - accuracy: 0.9174 - val_loss: 0.7038 - val_accuracy: 0.9310
Epoch 10/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7948 - accuracy: 0.9227
Epoch 00010: val_loss improved from 0.68513 to 0.67545, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-10-0.93.hdf5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.7948 - accuracy: 0.9227 - val_loss: 0.6755 - val_accuracy: 0.9342
Epoch 11/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7794 - accuracy: 0.9242
Epoch 00011: val_loss did not improve from 0.67545
187091/187091 [==============================] - 95s 507us/sample - loss: 0.7794 - accuracy: 0.9242 - val_loss: 0.7465 - val_accuracy: 0.9261
Epoch 12/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7231 - accuracy: 0.9297
Epoch 00012: val_loss improved from 0.67545 to 0.64818, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-12-0.94.hdf5
187091/187091 [==============================] - 95s 509us/sample - loss: 0.7231 - accuracy: 0.9297 - val_loss: 0.6482 - val_accuracy: 0.9367
Epoch 13/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6974 - accuracy: 0.9329
Epoch 00013: val_loss did not improve from 0.64818
187091/187091 [==============================] - 95s 507us/sample - loss: 0.6974 - accuracy: 0.9329 - val_loss: 1.3413 - val_accuracy: 0.8625
Epoch 14/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7894 - accuracy: 0.9233
```

```
Epoch 00014: val_loss improved from 0.64818 to 0.64730, saving model to /content/drive/MyDrive/ML_2/project/weights-improvement-14-0.94.hdf5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.7894 - accuracy: 0.9233 - val_loss: 0.6473 - val_accuracy: 0.9370
Epoch 15/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6717 - accuracy: 0.9350
Epoch 00015: val_loss did not improve from 0.64730
187091/187091 [==============================] - 95s 507us/sample - loss: 0.6717 - accuracy: 0.9350 - val_loss: 0.7350 - val_accuracy: 0.9277
<tensorflow.python.keras.callbacks.History at 0x7f64145a4160>
```

```
model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 15,
        batch_size = 1200,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
Train on 187091 samples, validate on 62364 samples
Epoch 1/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5812 - accuracy: 0.9438
Epoch 00001: val_loss improved from inf to 0.57225, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 95s 508us/sample - loss: 0.5812 - accuracy: 0.9438 - val_loss: 0.5723 - val_accuracy: 0.9445
Epoch 2/15
187091/187091 [==============================] - ETA: 0s - loss: 0.7543 - accuracy: 0.9272
Epoch 00002: val_loss improved from 0.57225 to 0.56193, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 521us/sample - loss: 0.7543 - accuracy: 0.9272 - val_loss: 0.5619 - val_accuracy: 0.9457
Epoch 3/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6871 - accuracy: 0.9339
Epoch 00003: val_loss improved from 0.56193 to 0.54938, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 520us/sample - loss: 0.6871 - accuracy: 0.9339 - val_loss: 0.5494 - val_accuracy: 0.9472
Epoch 4/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6919 - accuracy: 0.9338
Epoch 00004: val_loss improved from 0.54938 to 0.54300, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 521us/sample - loss: 0.6919 - accuracy: 0.9338 - val_loss: 0.5430 - val_accuracy: 0.9481
Epoch 5/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6541 - accuracy: 0.9372
Epoch 00005: val_loss improved from 0.54300 to 0.53510, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 518us/sample - loss: 0.6541 - accuracy: 0.9372 - val_loss: 0.5351 - val_accuracy: 0.9487
Epoch 6/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6541 - accuracy: 0.9376
Epoch 00006: val_loss improved from 0.53510 to 0.53071, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 521us/sample - loss: 0.6541 - accuracy: 0.9376 - val_loss: 0.5307 - val_accuracy: 0.9493
Epoch 7/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5629 - accuracy: 0.9463
Epoch 00007: val_loss improved from 0.53071 to 0.50691, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 519us/sample - loss: 0.5629 - accuracy: 0.9463 - val_loss: 0.5069 - val_accuracy: 0.9517
Epoch 8/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5785 - accuracy: 0.9450
Epoch 00008: val_loss improved from 0.50691 to 0.49867, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 98s 521us/sample - loss: 0.5785 - accuracy: 0.9450 - val_loss: 0.4987 - val_accuracy: 0.9526
Epoch 9/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6008 - accuracy: 0.9431
Epoch 00009: val_loss did not improve from 0.49867
187091/187091 [==============================] - 95s 507us/sample - loss: 0.6008 - accuracy: 0.9431 - val_loss: 0.5346 - val_accuracy: 0.9489
Epoch 10/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6166 - accuracy: 0.9417
Epoch 00010: val_loss improved from 0.49867 to 0.48489, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 98s 522us/sample - loss: 0.6166 - accuracy: 0.9417 - val_loss: 0.4849 - val_accuracy: 0.9540
Epoch 11/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5479 - accuracy: 0.9482
Epoch 00011: val_loss improved from 0.48489 to 0.47174, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 521us/sample - loss: 0.5479 - accuracy: 0.9482 - val_loss: 0.4717 - val_accuracy: 0.9553
Epoch 12/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5703 - accuracy: 0.9463
```

```
Epoch 00012: val_loss improved from 0.47174 to 0.46077, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 98s 521us/sample - loss: 0.5703 - accuracy: 0.9463 - val_loss: 0.4608 - val_accuracy: 0.9565
Epoch 13/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5542 - accuracy: 0.9480
Epoch 00013: val_loss improved from 0.46077 to 0.45470, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 521us/sample - loss: 0.5542 - accuracy: 0.9480 - val_loss: 0.4547 - val_accuracy: 0.9572
Epoch 14/15
187091/187091 [==============================] - ETA: 0s - loss: 0.5631 - accuracy: 0.9472
Epoch 00014: val_loss improved from 0.45470 to 0.45435, saving model to /content/drive/MyDrive/ML_2/project/best_model.h5
187091/187091 [==============================] - 97s 520us/sample - loss: 0.5631 - accuracy: 0.9472 - val_loss: 0.4544 - val_accuracy: 0.9574
Epoch 15/15
187091/187091 [==============================] - ETA: 0s - loss: 0.6172 - accuracy: 0.9421
```

Saved 5 diff models after each batch run for loading and testing

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

Creating the latent space

```
Z_mean = model.encoder.predict(X_test)
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in Te
  warnings.warn('`Model.state_updates` will be removed in a future version. '
```

```
Z_mean.shape
```

```
(62364, 114)
```

```
Y_test.shape
```

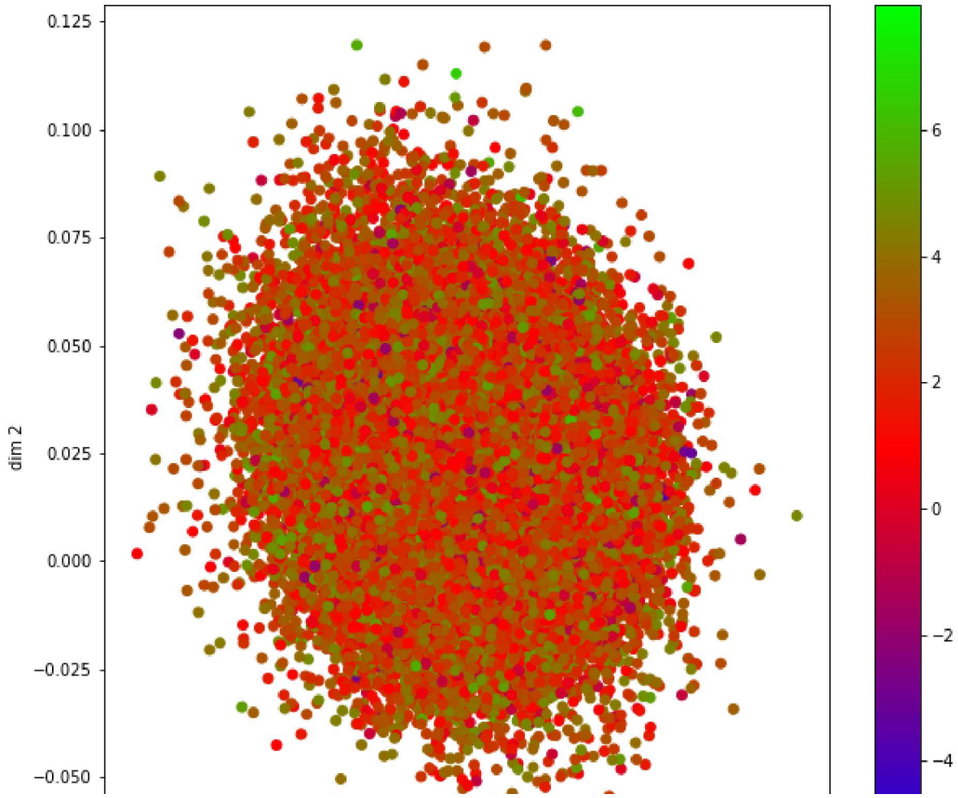
```
(62364, 114, 58)
```

```
from rdkit.Chem import Draw, Descriptors
```

```
logp = smiles_test.apply(Chem.MolFromSmiles).apply(Descriptors.MolLogP)
```

Plotting the latent space of molecules

```
#Plot dim1 and dim2 for mu
plt.figure(figsize=(10, 10))
plt.scatter(Z_mean[:, 0], Z_mean[:, 1],c=logp ,cmap='brg')
plt.xlabel('dim 1')
plt.ylabel('dim 2')
plt.colorbar()
plt.show()
```



Loading best saved model for generating molecules

```
model1 = MoleculeVAE()
```

```
trained_model = '/content/drive/MyDrive/ML_2/project/new_best_model.h5'
```

```
model1.load(charset, trained_model, latent_rep_size = 114)
```

```
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
```

```
test_sample = 'CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1'

test_sample
```

```
'CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1'
```

```
test_latent = encode_smiles(test_sample, model1, charset)
test_latent.size
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in Te
  warnings.warn('`Model.state_updates` will be removed in a future version. '
114
```

```
test_latent
```

```
array([[ 2.6943812e-01,  4.4802862e-01,  6.9935143e-01, -1.3222426e-01,
        -1.0009681e+00, -1.0238763e+00, -4.9647033e-01,  9.7381091e-01,
        -5.8185661e-01,  7.1797502e-01, -1.1016978e+00, -3.3689213e-01,
        -9.1568120e-02,  3.1968614e-01, -8.2250893e-02,  1.2293563e+00,
        -2.6550403e-01,  3.2556346e-03, -4.2492497e-01, -1.3270947e+00,
         7.1313989e-01,  1.4448118e+00, -2.2483233e-01, -2.1228179e-01,
        -9.5282108e-01,  1.5468325e-01, -4.3868098e-02, -4.1603562e-01,
         1.7389815e-01,  4.0020490e-01, -2.1981010e-01, -8.8252969e-02,
         1.1269593e+00, -3.2735610e-01,  1.7266257e-01, -7.9779774e-02,
        -7.7878034e-01,  7.8688401e-01,  3.7923267e-01, -6.2484551e-02,
        -5.7870579e-01, -5.4256804e-04,  1.3275197e-01, -6.3895985e-02,
        -1.0216132e+00, -6.2782842e-01, -6.1507430e-02, -3.0398117e-02,
        -6.0013634e-01, -2.3300819e-02, -1.0248888e+00, -8.8183290e-01,
        -6.6360348e-01, -3.0385172e-01, -3.5216865e-01, -1.8752995e-01,
        -8.5785352e-02, -1.2890995e-01,  1.0652457e+00, -1.0559481e+00,
        -1.1729375e+00, -8.4464562e-01, -6.9445133e-02, -6.3617630e-03,
         9.0211862e-01, -1.5108003e-01,  8.5416310e-02, -1.1813930e+00,
         4.1262487e-01,  7.9726738e-01, -1.0015085e+00,  5.5205595e-01,
        -1.3883901e-01,  3.1483808e-01, -4.4188207e-01,  2.6013759e-01,
        -6.0062510e-01,  8.4177589e-01,  7.4040920e-02, -1.7909527e-01,
        -7.2551541e-02,  1.1791281e-01,  7.7647798e-02, -1.2444323e+00,
         5.4230444e-02, -3.0431932e-01,  1.3318371e+00, -4.8001993e-01,
         1.8814114e-01,  1.2882185e-01,  3.0627063e-01, -3.3506840e-01,
         3.5230038e-01,  9.7244489e-01,  1.0294955e+00, -8.0255318e-01,
        -2.5572041e-02,  1.2923624e-01,  2.0635755e-01,  8.2585558e-02,
        -4.8638549e-01, -7.2807819e-04,  5.6422967e-01, -1.1829750e-01,
        -1.3363318e-01,  1.2784027e+00, -6.5891407e-02,  1.2261010e+00,
        -3.8502389e-01,  9.2760533e-01,  1.0739330e-01,  3.0039094e-02,
         2.1217023e-01,  5.2365369e-01]], dtype=float32)
```

```
latent_dim = 114
```

```
reconstructed_test = decode_latent_molecule(test_latent,model1,charset,latent_dim)
original = Chem.MolFromSmiles(test_sample)

reconstructed_test
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in Te
  warnings.warn('`Model.state_updates` will be removed in a future version. '
')(C\n\nc=Nooc\n)(CC\nc!o)(#CI#-oCI@C\n\n)(#Co-#CC#-oCCE@@)))#C)(#C)))C)EEEEEEEEEE))))))))))))))))ooooEEEE@@OOOOOOOOOO'
```

The reconstructed molecule doesnt give good results as it doesnt resemble the original model

```
paracetamol_smiles = 'CC(=O)Nc1ccc(O)cc1'
celecoxib_smiles = 'Cc1ccc(cc1)c2cc(nn2c3ccc(cc3)S(=O)(=O)N)C(F)(F)F'

steps = 10
results = interpolate(paracetamol_smiles, celecoxib_smiles, steps, charset, model1, latent_dim)

for result in results:
  print(result)
```

```
#
)(#C(#s)(##C(#C)(#C(#C-#-CC)-#CI#--C)(#C))(CC))CCC@CE@@EEE))))))))FFF)))))II)))EEEEEEEOOOOOOOOOO
)
c
```

```
Ic=N!o
6@

)(#C(#C)(#C)(#C)(#C(#--oCI#-oC)##-oC)(#C)(#C))#C)@CE@@EEEEE))))))))))))))))))))EEEEEEEE000000000
@)
c
Ic=N!o
(@

)(#C(sC)(#C)(#C
)(#CI#-oCI#--C)(#-c)((CC)(#C))#C)ECE@EEEEEE))))))))))))))III)ooEEEEEEE000000000
@)
c
Ic=N!o
(@C
)(#C((C)(#C6)
C))((#C--oCI#--C)(#-oC)(#C)(#C)))C)EEEEEEEEEE))))))))))))))IIIIooEEEEEEE000000000
@

c
Ic=!!o
(@

((#-ssC(##C66
C))(##C--oCI#--C)(#-oC)(#C)(#C)))C)EEEEEEEEEEE))))))))))))IIIIIIoEEEEEEE000000000
@




c!!oo(
C
)(#CI--C)(#C
@
)(#CI--C)(#C
@

[)(#CE--oCC#--oCI#-oC)(#C)(#C))))EEEEEEEEEEE))))))))))))IIIIIIoEEEEEEE000000000
@



cc!!ooN(C
)(#C#--C)(#C
6


)(#CE@C-(#C)--oCI@--c)(#C))#C)))EEEEEEEEEEE))))))))))))IIIIIIoEEEEEEE000000000
@



c!!oNNNCC)(#CI#-oC)(#C

N!oo[(CE@@
(oC)--oCI@#-c)(#C))#C)))EEEEEEEEEEE))))))))))))IIIIIIEEEEEEE]00000000
```

```
model.create(charset, latent_rep_size)

checkpointer = ModelCheckpoint(filepath = '/content/drive/MyDrive/ML_2/project/new_best_model.h5',
                               verbose = 1,
                               save_best_only = True)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
```

```
                                min_lr = 0.0001)

model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 5,
        batch_size = 1000,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_2 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
WARNING:tensorflow:Layer gru_3 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
Train on 187091 samples, validate on 62364 samples
Epoch 1/5
187091/187091 [==============================] - ETA: 0s - loss: 5.2616 - accuracy: 0.6580/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.
  warnings.warn('`Model.state_updates` will be removed in a future version. '

Epoch 00001: val_loss improved from inf to 4.46655, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 108s 579us/sample - loss: 5.2616 - accuracy: 0.6580 - val_loss: 4.4665 - val_accuracy: 0.6793
Epoch 2/5
187091/187091 [==============================] - ETA: 0s - loss: 4.2398 - accuracy: 0.6887
Epoch 00002: val_loss improved from 4.46655 to 4.07643, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 513us/sample - loss: 4.2398 - accuracy: 0.6887 - val_loss: 4.0764 - val_accuracy: 0.6938
Epoch 3/5
187091/187091 [==============================] - ETA: 0s - loss: 4.0968 - accuracy: 0.6958
Epoch 00003: val_loss improved from 4.07643 to 3.99533, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 97s 516us/sample - loss: 4.0968 - accuracy: 0.6958 - val_loss: 3.9953 - val_accuracy: 0.7065
Epoch 4/5
187091/187091 [==============================] - ETA: 0s - loss: 4.0364 - accuracy: 0.7062
Epoch 00004: val_loss improved from 3.99533 to 3.91950, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 97s 518us/sample - loss: 4.0364 - accuracy: 0.7062 - val_loss: 3.9195 - val_accuracy: 0.7126
Epoch 5/5
187091/187091 [==============================] - ETA: 0s - loss: 3.9544 - accuracy: 0.7134
Epoch 00005: val_loss improved from 3.91950 to 3.91662, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 97s 516us/sample - loss: 3.9544 - accuracy: 0.7134 - val_loss: 3.9166 - val_accuracy: 0.7138
<tensorflow.python.keras.callbacks.History at 0x7f6408d047f0>
```

```
model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 20,
        batch_size = 1200,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
Epoch 00006: val_loss improved from 3.55235 to 3.37188, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 3.3718 - accuracy: 0.7514 - val_loss: 3.3719 - val_accuracy: 0.7526
Epoch 7/20
187091/187091 [==============================] - ETA: 0s - loss: 3.2337 - accuracy: 0.7610
Epoch 00007: val_loss improved from 3.37188 to 3.16517, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 3.2337 - accuracy: 0.7610 - val_loss: 3.1652 - val_accuracy: 0.7657
Epoch 8/20
187091/187091 [==============================] - ETA: 0s - loss: 3.1337 - accuracy: 0.7696
Epoch 00008: val_loss improved from 3.16517 to 3.02814, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 3.1337 - accuracy: 0.7696 - val_loss: 3.0281 - val_accuracy: 0.7775
Epoch 9/20
```

```
187091/187091 [==============================] - ETA: 0s - loss: 2.9810 - accuracy: 0.7824
Epoch 00009: val_loss improved from 3.02814 to 2.87562, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 2.9810 - accuracy: 0.7824 - val_loss: 2.8756 - val_accuracy: 0.7927
Epoch 10/20
187091/187091 [==============================] - ETA: 0s - loss: 2.9211 - accuracy: 0.7873
Epoch 00010: val_loss improved from 2.87562 to 2.84089, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 2.9211 - accuracy: 0.7873 - val_loss: 2.8409 - val_accuracy: 0.7955
Epoch 11/20
187091/187091 [==============================] - ETA: 0s - loss: 2.8168 - accuracy: 0.7961
Epoch 00011: val_loss improved from 2.84089 to 2.75506, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 509us/sample - loss: 2.8168 - accuracy: 0.7961 - val_loss: 2.7551 - val_accuracy: 0.7999
Epoch 12/20
187091/187091 [==============================] - ETA: 0s - loss: 2.7408 - accuracy: 0.8025
Epoch 00012: val_loss improved from 2.75506 to 2.63515, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 2.7408 - accuracy: 0.8025 - val_loss: 2.6352 - val_accuracy: 0.8129
Epoch 13/20
187091/187091 [==============================] - ETA: 0s - loss: 2.6699 - accuracy: 0.8092
Epoch 00013: val_loss improved from 2.63515 to 2.52175, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 510us/sample - loss: 2.6699 - accuracy: 0.8092 - val_loss: 2.5218 - val_accuracy: 0.8241
Epoch 14/20
187091/187091 [==============================] - ETA: 0s - loss: 2.5708 - accuracy: 0.8175
Epoch 00014: val_loss did not improve from 2.52175
187091/187091 [==============================] - 93s 496us/sample - loss: 2.5708 - accuracy: 0.8175 - val_loss: 2.5942 - val_accuracy: 0.8168
Epoch 15/20
187091/187091 [==============================] - ETA: 0s - loss: 2.4891 - accuracy: 0.8246
Epoch 00015: val_loss improved from 2.52175 to 2.48962, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 512us/sample - loss: 2.4891 - accuracy: 0.8246 - val_loss: 2.4896 - val_accuracy: 0.8225
Epoch 16/20
187091/187091 [==============================] - ETA: 0s - loss: 2.3778 - accuracy: 0.8351
Epoch 00016: val_loss improved from 2.48962 to 2.25999, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 512us/sample - loss: 2.3778 - accuracy: 0.8351 - val_loss: 2.2600 - val_accuracy: 0.8448
Epoch 17/20
187091/187091 [==============================] - ETA: 0s - loss: 2.2624 - accuracy: 0.8438
Epoch 00017: val_loss improved from 2.25999 to 2.21004, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 2.2624 - accuracy: 0.8438 - val_loss: 2.2100 - val_accuracy: 0.8491
Epoch 18/20
187091/187091 [==============================] - ETA: 0s - loss: 2.1762 - accuracy: 0.8509
Epoch 00018: val_loss improved from 2.21004 to 2.05551, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 2.1762 - accuracy: 0.8509 - val_loss: 2.0555 - val_accuracy: 0.8601
Epoch 19/20
187091/187091 [==============================] - ETA: 0s - loss: 2.0812 - accuracy: 0.8587
Epoch 00019: val_loss improved from 2.05551 to 1.88248, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 511us/sample - loss: 2.0812 - accuracy: 0.8587 - val_loss: 1.8825 - val_accuracy: 0.8747
Epoch 20/20
187091/187091 [==============================] - ETA: 0s - loss: 2.0068 - accuracy: 0.8650
Epoch 00020: val_loss did not improve from 1.88248
187091/187091 [==============================] - 93s 497us/sample - loss: 2.0068 - accuracy: 0.8650 - val_loss: 2.2958 - val_accuracy: 0.8435
<tensorflow.python.keras.callbacks.History at 0x7f64083e4668>
```

```
model.autoencoder.fit(
         X_train,
         Y_train,
         shuffle = True,
         epochs = 20,
         batch_size = 1200,
        callbacks = [checkpointer, reduce_lr],
        validation_data = (X_test,Y_test))
```

```
Train on 187091 samples, validate on 62364 samples
Epoch 1/20
187091/187091 [==============================] - ETA: 0s - loss: 1.8985 - accuracy: 0.8731
Epoch 00001: val_loss improved from 1.88248 to 1.78969, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 93s 499us/sample - loss: 1.8985 - accuracy: 0.8731 - val_loss: 1.7897 - val_accuracy: 0.8818
Epoch 2/20
```

```
187091/187091 [==============================] - ETA: 0s - loss: 1.8292 - accuracy: 0.8778
Epoch 00002: val_loss improved from 1.78969 to 1.71855, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 1.8292 - accuracy: 0.8778 - val_loss: 1.7185 - val_accuracy: 0.8862
Epoch 3/20
187091/187091 [==============================] - ETA: 0s - loss: 1.7115 - accuracy: 0.8865
Epoch 00003: val_loss did not improve from 1.71855
187091/187091 [==============================] - 93s 497us/sample - loss: 1.7115 - accuracy: 0.8865 - val_loss: 2.7508 - val_accuracy: 0.8281
Epoch 4/20
187091/187091 [==============================] - ETA: 0s - loss: 1.7311 - accuracy: 0.8856
Epoch 00004: val_loss improved from 1.71855 to 1.55610, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 512us/sample - loss: 1.7311 - accuracy: 0.8856 - val_loss: 1.5561 - val_accuracy: 0.8978
Epoch 5/20
187091/187091 [==============================] - ETA: 0s - loss: 1.6098 - accuracy: 0.8939
Epoch 00005: val_loss improved from 1.55610 to 1.49812, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 512us/sample - loss: 1.6098 - accuracy: 0.8939 - val_loss: 1.4981 - val_accuracy: 0.9027
Epoch 6/20
187091/187091 [==============================] - ETA: 0s - loss: 1.5050 - accuracy: 0.9014
Epoch 00006: val_loss improved from 1.49812 to 1.43657, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 1.5050 - accuracy: 0.9014 - val_loss: 1.4366 - val_accuracy: 0.9050
Epoch 7/20
187091/187091 [==============================] - ETA: 0s - loss: 1.5141 - accuracy: 0.9011
Epoch 00007: val_loss did not improve from 1.43657
187091/187091 [==============================] - 93s 496us/sample - loss: 1.5141 - accuracy: 0.9011 - val_loss: 1.4737 - val_accuracy: 0.9016
Epoch 8/20
187091/187091 [==============================] - ETA: 0s - loss: 1.4919 - accuracy: 0.9027
Epoch 00008: val_loss did not improve from 1.43657
187091/187091 [==============================] - 93s 495us/sample - loss: 1.4919 - accuracy: 0.9027 - val_loss: 1.9693 - val_accuracy: 0.8679
Epoch 9/20
187091/187091 [==============================] - ETA: 0s - loss: 1.4393 - accuracy: 0.9057
Epoch 00009: val_loss improved from 1.43657 to 1.23562, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 1.4393 - accuracy: 0.9057 - val_loss: 1.2356 - val_accuracy: 0.9197
Epoch 10/20
187091/187091 [==============================] - ETA: 0s - loss: 1.4545 - accuracy: 0.9049
Epoch 00010: val_loss did not improve from 1.23562
187091/187091 [==============================] - 93s 495us/sample - loss: 1.4545 - accuracy: 0.9049 - val_loss: 1.2984 - val_accuracy: 0.9161
Epoch 11/20
187091/187091 [==============================] - ETA: 0s - loss: 1.3206 - accuracy: 0.9143
Epoch 00011: val_loss improved from 1.23562 to 1.21221, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 1.3206 - accuracy: 0.9143 - val_loss: 1.2122 - val_accuracy: 0.9213
Epoch 12/20
187091/187091 [==============================] - ETA: 0s - loss: 1.2932 - accuracy: 0.9162
Epoch 00012: val_loss did not improve from 1.21221
187091/187091 [==============================] - 93s 497us/sample - loss: 1.2932 - accuracy: 0.9162 - val_loss: 1.3036 - val_accuracy: 0.9149
Epoch 13/20
187091/187091 [==============================] - ETA: 0s - loss: 1.1753 - accuracy: 0.9239
Epoch 00013: val_loss improved from 1.21221 to 1.06654, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 510us/sample - loss: 1.1753 - accuracy: 0.9239 - val_loss: 1.0665 - val_accuracy: 0.9314
Epoch 14/20
187091/187091 [==============================] - ETA: 0s - loss: 1.3141 - accuracy: 0.9154
Epoch 00014: val_loss improved from 1.06654 to 1.04924, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 95s 509us/sample - loss: 1.3141 - accuracy: 0.9154 - val_loss: 1.0492 - val_accuracy: 0.9326
Epoch 15/20
187091/187091 [==============================] - ETA: 0s - loss: 1.2458 - accuracy: 0.9196
```

```
model.autoencoder.fit(
        X_train,
        Y_train,
        shuffle = True,
        epochs = 20,
        batch_size = 1200,
       callbacks = [checkpointer, reduce_lr],
       validation_data = (X_test,Y_test))
```

```
Train on 187091 samples, validate on 62364 samples
```

```
Epoch 1/20
187091/187091 [==============================] - ETA: 0s - loss: 0.9878 - accuracy: 0.9374
Epoch 00001: val_loss improved from 0.89889 to 0.87196, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 96s 513us/sample - loss: 0.9878 - accuracy: 0.9374 - val_loss: 0.8720 - val_accuracy: 0.9447
Epoch 2/20
187091/187091 [==============================] - ETA: 0s - loss: 1.0080 - accuracy: 0.9361
Epoch 00002: val_loss improved from 0.87196 to 0.84889, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 99s 529us/sample - loss: 1.0080 - accuracy: 0.9361 - val_loss: 0.8489 - val_accuracy: 0.9463
Epoch 3/20
187091/187091 [==============================] - ETA: 0s - loss: 1.0296 - accuracy: 0.9351
Epoch 00003: val_loss improved from 0.84889 to 0.82698, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 98s 526us/sample - loss: 1.0296 - accuracy: 0.9351 - val_loss: 0.8270 - val_accuracy: 0.9479
Epoch 4/20
187091/187091 [==============================] - ETA: 0s - loss: 1.1801 - accuracy: 0.9258
Epoch 00004: val_loss did not improve from 0.82698
187091/187091 [==============================] - 96s 514us/sample - loss: 1.1801 - accuracy: 0.9258 - val_loss: 1.2268 - val_accuracy: 0.9201
Epoch 5/20
187091/187091 [==============================] - ETA: 0s - loss: 0.9452 - accuracy: 0.9405
Epoch 00005: val_loss improved from 0.82698 to 0.79311, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 99s 529us/sample - loss: 0.9452 - accuracy: 0.9405 - val_loss: 0.7931 - val_accuracy: 0.9503
Epoch 6/20
187091/187091 [==============================] - ETA: 0s - loss: 1.0275 - accuracy: 0.9358
Epoch 00006: val_loss did not improve from 0.79311
187091/187091 [==============================] - 96s 513us/sample - loss: 1.0275 - accuracy: 0.9358 - val_loss: 0.8181 - val_accuracy: 0.9481
Epoch 7/20
187091/187091 [==============================] - ETA: 0s - loss: 0.9636 - accuracy: 0.9400
Epoch 00007: val_loss did not improve from 0.79311
187091/187091 [==============================] - 96s 512us/sample - loss: 0.9636 - accuracy: 0.9400 - val_loss: 1.5653 - val_accuracy: 0.9095
Epoch 8/20
187091/187091 [==============================] - ETA: 0s - loss: 0.8478 - accuracy: 0.9470
Epoch 00008: val_loss improved from 0.79311 to 0.73927, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 98s 526us/sample - loss: 0.8478 - accuracy: 0.9470 - val_loss: 0.7393 - val_accuracy: 0.9540
Epoch 9/20
187091/187091 [==============================] - ETA: 0s - loss: 0.9591 - accuracy: 0.9409
Epoch 00009: val_loss did not improve from 0.73927
187091/187091 [==============================] - 96s 512us/sample - loss: 0.9591 - accuracy: 0.9409 - val_loss: 1.0296 - val_accuracy: 0.9346
Epoch 10/20
187091/187091 [==============================] - ETA: 0s - loss: 0.8681 - accuracy: 0.9461
Epoch 00010: val_loss improved from 0.73927 to 0.70448, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 98s 526us/sample - loss: 0.8681 - accuracy: 0.9461 - val_loss: 0.7045 - val_accuracy: 0.9563
Epoch 11/20
187091/187091 [==============================] - ETA: 0s - loss: 0.8409 - accuracy: 0.9479
Epoch 00011: val_loss improved from 0.70448 to 0.68244, saving model to /content/drive/MyDrive/ML_2/project/new_best_model.h5
187091/187091 [==============================] - 99s 527us/sample - loss: 0.8409 - accuracy: 0.9479 - val_loss: 0.6824 - val_accuracy: 0.9578
Epoch 12/20
187091/187091 [==============================] - ETA: 0s - loss: 1.0983 - accuracy: 0.9316
Epoch 00012: val_loss did not improve from 0.68244
187091/187091 [==============================] - 95s 510us/sample - loss: 1.0983 - accuracy: 0.9316 - val_loss: 0.7172 - val_accuracy: 0.9555
Epoch 13/20
187091/187091 [==============================] - ETA: 0s - loss: 0.8931 - accuracy: 0.9451
Epoch 00013: val_loss did not improve from 0.68244
187091/187091 [==============================] - 96s 512us/sample - loss: 0.8931 - accuracy: 0.9451 - val_loss: 0.6929 - val_accuracy: 0.9572
Epoch 14/20
187091/187091 [==============================] - ETA: 0s - loss: 0.9849 - accuracy: 0.9394
Epoch 00014: val_loss did not improve from 0.68244
187091/187091 [==============================] - 99s 530us/sample - loss: 0.9849 - accuracy: 0.9394 - val_loss: 0.7360 - val_accuracy: 0.9545
Epoch 15/20
187091/187091 [==============================] - ETA: 0s - loss: 0.6449 - accuracy: 0.9607
```

Trained the model again in batches to improve accuracy but the output from the trained model was not satisfactory trying LSTM based approach in another notebook to see the results