UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

---

de Novo Design of Small Molecules using Variational and Conditional Variational Autoencoders

---

Author:
Špela ZAVODNIK

Supervisor:
Dr. Jordi VITRIA MARCA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamentals of Data Science

in the

Facultat de Matemàtiques i Informàtica

September 2, 2019

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

de Novo Design of Small Molecules using Variational and Conditional Variational Autoencoders

by Špela ZAVODNIK

Chemical space is estimated to contain over 10^{60} small synthetically feasible molecules and so far only a fraction of the space has been explored. Experimental techniques are time-consuming and expensive so computational methods, such as machine learning, are needed for efficient exploration. Here we looked at generative models, more specifically variational autoencoder (VAE) and conditional variational autoencoder (CVAE), used for designing new molecules. In the first part, we evaluated already written VAE and in the second part, we upgraded it to the CVAE. For the conditional vectors in CVAE we used B4 Signatures generated from Chemical Checker describing molecular properties. Both models performed well, however, CVAE showed many advantages.

Acknowledgements

First and foremost I would like to thank my project supervisors, Dr. Jordi Vitrià, Miquel Duran-Frigola and Dr. Patrick Aloy for their help with the thesis.

Secondly, I'd like to thank everyone who put up with me during the past year complaining and to everyone who subtly told me to not stress out.

Contents

Chapter 1

Introduction

Machine Learning, especially its sub-field Deep Learning, has accomplished exceptional results in various areas of scientific research during recent years due to its broad applicability. Chemoinformatics is just one of the disciplines that has benefited greatly from the use of machine learning algorithms. While there are many areas of chemoinformatics where artificial intelligence produced revolutionary results, *de novo* molecular design has been wholly transformed. The use of modern algorithms for discovering new molecules speeds up the process, is way more efficient than traditional methods and it also offers more pliability.

The number of potential organic molecules[1] is enormous even when we take into account only small molecules that are synthetically feasible and are not heavier than 500 Dalton[2]. The chemical space of such molecules is estimated to contain over 10^{60} unique members (Virshup et al., 2013), from which only a fraction of a percentage have been identified and thus far only $\sim 10^8$ substances been synthesized (Kim et al., 2015), leaving more than 99% of chemical space unexplored.

Until recently quite a large number of small molecules used for medical treatments were natural products[3] or molecules derived from them (Newman and Cragg, 2016). The rest were mainly discovered through trial and error and by using educated guesses. The main challenge of drug discovery is to generate novel compounds with desired properties (e.g. molecular weight, toxicity, solubility, etc.) and with novel bioactivity[4] profiles. Creating such molecules experimentally is time consuming and inefficient thus machine learning is necessary for exploration of the unfamiliar regions of the chemical space. Machine Learning is a much more robust approach offering efficient search that optimizes the available resources and accelerates the drug discovery pipeline.

[1] Molecules that contain mainly carbon (C) and hydrogen (H) atoms as well as nitrogen (N), sulfur (S), phosphorus (P) and oxygen (O).

[2] $\frac{1}{12}$ of the mass of an unbound neutral atom of carbon-12 in its nuclear and electronic ground state and at rest.

[3] drugs isolated from natural materials such as plants or are produced by microbes etc.

[4] Bioactive compounds interact with organism consuming them on cellular level

1.1 Motivation

The aim of this thesis is to investigate the use of variational autoencoders (VAE) and conditional variational autoencoders (CVAE) in terms of their capabilities of designing novel drug-like molecules. VAEs and CVAEs are generative models, that are capable of generating new data that resembles the input data. They convert discrete representation of molecules into a continuous representation - latent space - which can be easily manipulated.

Ideally a trained VAE should be able to generate valid molecules that are similar, but not too similar to the input molecule while CVAE should generate molecules following the input conditions. For evaluation of the results, similarity metrics and comparison of molecular properties will be used.

In the first part I used already trained VAE and preformed analysis on specific molecules to see how the model performs. I then upgraded this model to the CVAE, repeat the analysis and compared results with VAE.

Chapter 2

Molecular Representation

The format of the data used with machine learning algorithms is quite important as the algorithm needs to learn from it efficiently. Molecules can be represented in many different formats such as Coulomb matrix, 3D geometry, electronic density, SMILES, fingerprints, InChl keys[1] etc. The more detailed representation the more computationally expensive it will be to process it. While none of the molecular representations currently stores all of the molecular properties, each of the formats is useful for specific task, for example, SMILES representation is currently preferred format for generative models, although graph based models have become a popular choice for designing novel molecules in the past couple of years as well.

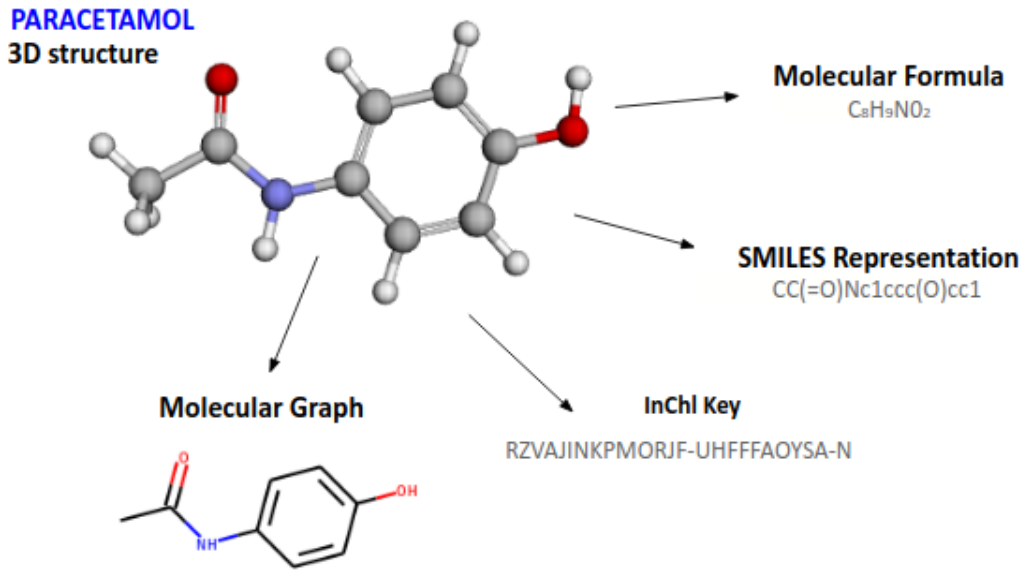

FIGURE 2.1: Paracetamol: Examples of its representations.

[1] Another string based representation that encodes more complex information than SMILES.

2.1 SMILES

The simplified molecular-input line-entry system (SMILES) is a one dimensional, line representation of a two dimensional chemical drawing. SMILES is a language with not many grammar rules and simple vocabulary that includes atom and bond symbols only. Not only is this linguistic notation way more compact than other representations, it also makes the use of natural language processing algorithms possible.

A molecule usually has more than one valid SMILES notation (SEE FIGURE - SMILES can be written starting from any atom of a molecule) so canonicalization algorithm, an algorithm that assigns one specific SMILES to each molecule, is normally applied. SMILES represent only a subset of possible molecules meaning that a syntactically non valid SMILES might still represent a valid molecular structure, but its physics is not taken into account by the set grammar rules although the number of such smiles seems to be negligible.

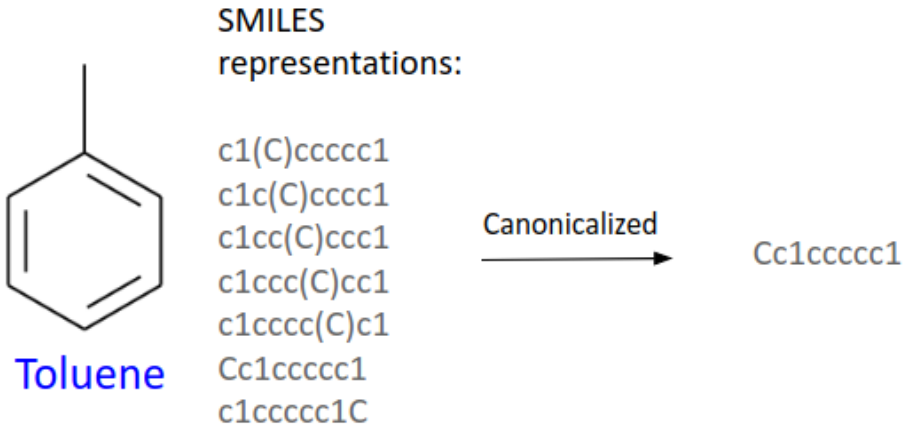

FIGURE 2.2: Molecular graph of Toluene and all of its SMILES representation.

2.2 Molecular Fingerprints

Fingerprints are a way of representing molecular structure as mathematical objects — typically vectors that correspond to the extracted molecular features. The features represented by the fingerprints depend on which type of a fingerprint was used. To evaluate the results of the VAE and CVAE we used Morgan Fingerprints.

Morgan Fingerprints

Morgan fingerprints also known as circular fingerprints store information about molecular connectivity. They look at the surroundings of each atom and store information about connectivity up to a given radius(neighbouring atoms, usually from 0-3) encoding each possible connection. The results are compressed to a bit vector by using hash algorithm. Each fingerprint bit corresponds to a fragment of the molecule.

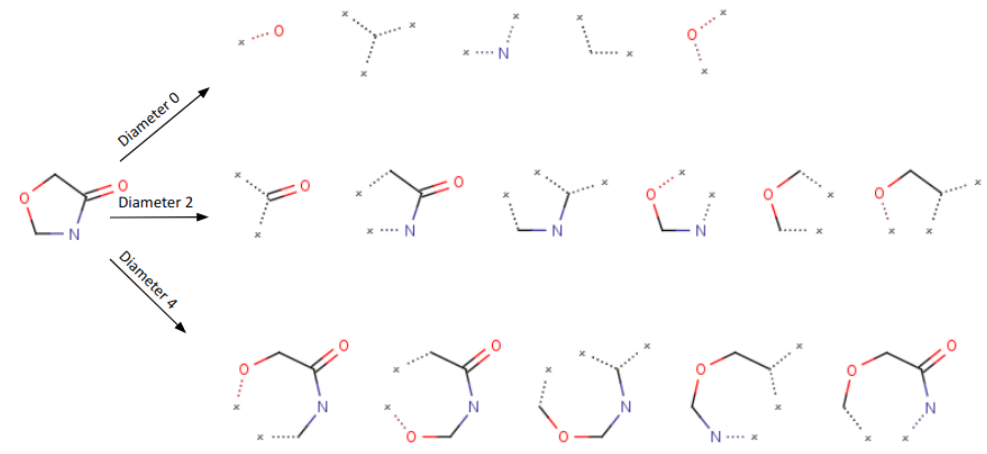

FIGURE 2.3: Scheme of Morgan fingerprinting for different diameters. Each one of the images is encoded into the fingerprint.

2.2.1 Tanimoto Similarity

There are several similarity and distance metrics methods that quantify the similarity between different molecular fingerprints.

Tanimoto similarity of molecule **a** and molecule **b** is calculated by using the following formula:

$$d(\mathbf{a}, \mathbf{b}) = \frac{n(A \cap B)}{n(A) + n(B) - (A \cap B)} \tag{2.1}$$

where:

$n(A)$ = number of bits set to 1 in molecule a's fingerprint.
$n(B)$ = number of bits set to 1 in molecule b's fingerprint.
$(A \cap B)$ = number of bits set to 1 that molecule **a**

and molecule **b** have in common.
As we can see Tanimoto normalizes the degree of size in the denominator.

The higher the similarity the more similar the molecular fingerprints. The values of similarity will be anywhere from 0 to 1, where higher the number the more bits have the molecular fingerprints in common. It's good to note that value of 1 does not mean the molecules are identical. Value of 1 only implies that they have identical structural descriptors fingerprint. Moreover, smaller molecules will often have lower similarity using Tanimoto coefficient.

Why is similarity important?

When scientists are looking for new drugs they usually start with already known drug and try to adopt them for other treatments, or they look for molecules in a database that are similar to the query molecule. The molecules that are similar to each other tend to have quite similar properties meaning that we would like to create molecules that are different from each other but still fairly alike. While Tanimoto distance is widely used to compare molecular fingerprints as it incorporates substructure matching and while is not the only similarity search metric it is one of the best for fingerprint-based similarity comparison (Bajusz, Rácz, and Héberger, 2015).

Chapter 3

Designing Molecules

The design of new molecules is categorized into two different approaches. The first one is a direct approach where the new molecules are found by experimentation or simulation. This approach starts in the chemical space and then moves onto determining the quantum properties[1] of the molecules. The exact quantum properties are calculated using the Schrödinger equation, which is computationally expensive. This problem is partly solved by simply approximating the properties, however, usually there are numerous molecules that we would like to analyze. The use of computational methods can cut down the costs by looking for specific molecules to get a smaller subset that is then analyzed further.

The Inverse design starts with the desired properties and looks for molecular structure within the chemical space that fit them. The two methods that use the inverse approach are high-throughput virtual screening and modern computational methods; optimisation and generative models.

3.1 Virtual Screening

High-throughput virtual screening, takes the initial library of candidates and filters them by their properties (such as solubility, toxicity, activity, etc.) which gives an indication of closeness between the molecules. Only the ones that fit the initial criteria, if there even are any, are then tested experimentally (Lionta et al., 2014; McInnes, 2007). The molecular structure of filtered molecules can then be modified to fit the desired properties using genetic algorithms and optimisation techniques, but, this process is highly inefficient (Rupakheti et al., 2015; Hartenfeller and Schneider, 2011; Schneider, 2018).

Contrarily, generative models aim to directly create novel molecules that fit wanted properties.

[1] Quantum mechanics allows us to predict properties of elements and how they behave in molecular structures.

3.2 Generative Models

3.2.1 Recurrent Neural Networks

Configuration of molecules can be expressed with SMILES strings (see subsection 2.1). New SMILES and hence new molecules can, therefore, be generated using natural language processing models such as recurrent neural networks (RNNs) that are used for data of sequential nature. RNNs learn the probability distribution of the order of characters in these strings.

In particular, much of this success has been achieved by the use of recurrent networks of long short-term memory (LSTM) cells that are very efficient and very precise at writing new strings. LSTMs are well-suited as they can process the entire sequences of data and not only single data points. When the RNN model is given a piece of a SMILES it will predict the probabilities of having any of the stated characters placed after the said set of already given characters, producing new SMILES strings.

Bjerrum and Threlfall (2017) explored how well can RNNs with LSTMs learn the grammatical rules of SMILES and generate synthesizable molecules. The molecules used for training and the novel molecules written by the model had very similar distributions of their properties (molecular weight, predicted logP[2]). The majority of the compounds created even had a good synthetic accessibility score[3]. Similar results were seen by Segler et al. (2017). The team also fine-tuned the model using a subset of molecules to perform transfer learning and enabled the creation of molecules with wanted activities and Gupta et al. (2018) mentioned that the fine-tuned molecules showed significantly higher similarity to the training set than the molecules that were not fine-tuned.

3.2.2 Autoencoders and Generative Adversarial Networks

Generative models have strongly influenced chemical design as they arguably could fully automate the search for new compounds. Advance types of autoencoders, variational autoencoders and conditional variational autoencoders, as well as generative adversarial networks (GANs) have already shown tremendous potential.

Autoencoders aim to recreate their inputs to their outputs by compressing the

[2]is a partition coefficient between octanol and water—very important property in drug design as it tells us how soluble a molecule is in water.

[3]score that tells us how easy or hard it is to synthesize a molecule.

input into a latent-space representation and decoding the output from this representation. In theory similar inputs are expected to have similar latent space representations and should be near each other. Team (Gómez-Bombarelli et al., 2018), experimented with VAE by using continuous optimisation to produce new compounds. Their autoencoder was trained using SMILES and their results showed that similar molecules are indeed found close together in the latent space. The VAE was also trained jointly with an additional neural network for predicting the properties of generated molecules.

To further improve VAE for molecular design, Kusner, Paige, and Hernández-Lobato (2017) wrote Grammar Variational Autoencoder (GrammarVAE) to solve the problem of sampled data points being invalid when transformed to their SMILES representation. They provided the VAE with explicit SMILES grammar rules that were available during the generation process. Their model did not have to learn grammar rules, but only semantic properties of the data. GrammarVAE generated more valid outputs and the latent space was more logical which resulted in nearby points being decoded to similar discrete outputs. Furthermore, Bjerrum and Sattarov (2018) also concluded that properties of the latent space are influenced by the choice of molecular representation.

VAEs generate new molecules by sampling the space around the query molecules. The generated molecules are similar and so are their properties. While this is a good strategy to find alike molecules we cannot use it to find molecules that have the exact properties that we want. This problem can be solved by using CVAE. Because molecular properties are correlated with each other, change in one might create a change in the other property which is not what we normally want. with CVAE we can control several properties at the same time. Lim et al. (2018) created a CVAE that was able to generate drug-like molecules that fit five target properties concurrently within an error range of 10%. The team also reported that it is possible to change one target without other 4 being influenced and that properties can be increased above the limits of the training set.

Blaschke et al. (2018) compared four different AE architectures for sequence generation. Their results not only depended on the architecture used, but also on the distribution of latent vectors of the encoder. Their best performing autoencoder was a Uniform adversarial autoencoder (AAE) — autoencoder that imposes a uniform distribution onto the latent representation of the autoencoder. Kadurin et al. (2017) created AAE and looked at its advantages compared to VAE. The results were compared on the basis of the reconstruction error and by comparing the changes between the molecular fingerprints. They have concluded that AAE enhanced the capacity and efficiency of creation of molecular structures with specific properties. De Cao and Kipf (2018) adapted GAN to use graph data and combined it with reinforcement learning to generate the molecules with desired chemical properties. Almost all of

the generated structures were valid.

3.2.3 Graph Models

As seen, most of the research on generating new compounds has been focused on using SMILES strings as a representation of molecules. Generative graph models, on the other hand, use graphs as inputs. Graph based generative models, outperform SMILES based models in many metrics, especially in the rate of valid outputs. Simonovsky and Komodakis (2018) created a VAE that uses molecular graphs as its inputs, but the GraphVAE only showed good results when generating smaller and less complex graphs.

Li, Zhang, and Liu (2018) employed conditional graph generative model that was effective at producing compounds similar to their input molecule and Li et al. (2018) have created a graph generative model that performed much better than Grammar-VAE, CVAE and GraphVAE.

Chapter 4

Neural Networks

As we will see in the next chapter, autoencoders are built using neural networks. The ones used in this thesis, include recurrent neural networks (RNNs), more specifically, gated recurrent units (GRUs) and 1D convolutional networks (1D CNNs), which will be the focus of this chapter.

4.1 Recurrent Neural Networks - RNNS

The main idea behind RNNs is to be able to make sense of sequential information. In a traditional neural network all inputs, as well as the outputs, are assumed to be independent of each other. However, sometimes we have tasks, such as generation of SMILES, where the sequence of characters is important. We can think of RNNs as multiple copies of the same network, passing the information forward.

RNNs perform the same task for every element of a sequence, with the output being depended on the previous computations. Their loopy architecture passes information from one step of the network to the next, so at every step, the network already has all of the information up until then.

Architecture of RNNs

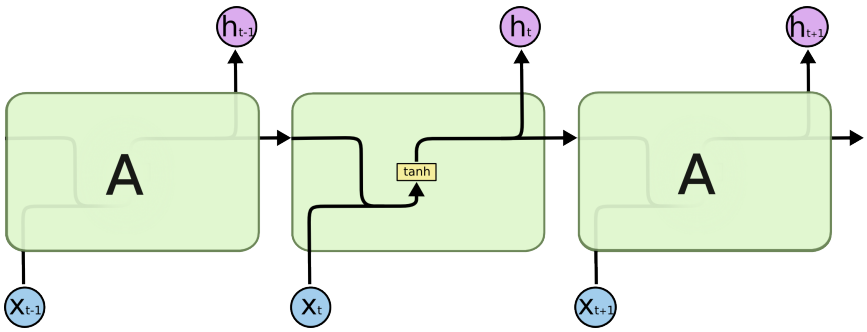

FIGURE 4.1: Architecture of RNN

Recurrent neural networks have the form of a chain of repeating cells of neural network. In standard RNNs, this cell will have quite a simple structure (see Figure 4.1) that shows the structure of the network and its cells.

Mathematical description of RNNs:

$$h_t = \sigma_h(U_h x_t + V_h h_{t-1} + b_h)$$
$$o_t = \sigma_y(W_y h_t + b_h)$$

Where:

x_t = input vector $(m \times 1)$
h_t = hidden vector $(n \times 1)$
o_t = output vector $(n \times 1)$
b_h = bias vector $(n \times 1)$
U, W = parameter matrices $(n \times m)$
V = parameter matrix $(n \times n)$
σ_h, σ_y = activation functions

4.1.1 Gated Recurrent Unit - GRU

GRUs were introduced by Cho et al. (2014). The main idea behind them is that they are able to solve the vanishing gradient problem that comes with standard RNNs. Gradient is a value that is used to update weights of the neural network. In RNNs, gradient shrinks as it back propagates through time and it does not contribute much to learning once the values become small. Layers that get small gradient update and that do not learn are usually the earlier layers. Because this layers do not learn, RNNs suffer from memory loss.

Unlike vanilla RNNs, GRUs are trained to keep information from steps that happened a while back and if the information is not needed they are able to forget it. They have an internal mechanism, Forget gate and Input gate, that solves this problem. These gates are regulating the flow of the information, they can learn which data in a sequence will they forget and which data is important for making predictions.

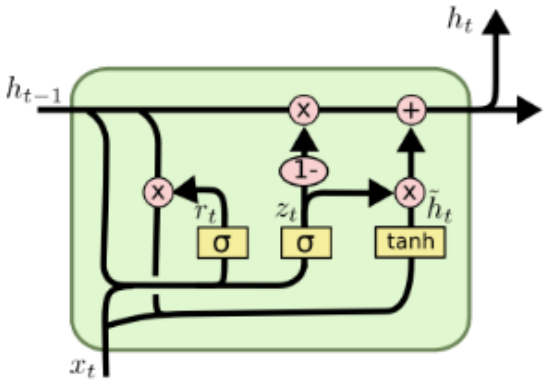

FIGURE 4.2: Architecture of GRU

Mathematical description of GRUs:

$$
\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\tilde{h}_t &= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

Where:

h_t = hidden layer vectors.
x_t = output vector.
b_z, b_r, b_h = bias vector.
W_z, W_r, W_h = parameter matrices.
σ, tanh = activation functions.

4.1.2 LSTMs

LSTMs (Hochreiter and Schmidhuber, 1997) are more complicated version of GRUs, or rather GRUs are a simpler version of LSTMs as they were developed years after. Just like GRUs, LSTMs avoid the vanishing gradient problem and can learn tasks that require remembering events that happened many steps earlier.

One of the main differences between the two is the number of gates and their role. LSTM has three gates while GRU has only two. The GRU's Update gate is

similar to the Input and Forget gates in the LSTM, the new memory content is controlled differently. Another difference between the two is that GRU stores log-term dependencies and short-term memory in a single hidden state while LSTM stores short-term memory in the hidded state and log-teerm dependencies in the cell state.

Due to fewer number of gates, GRUs have to update less weights and parameters during the training and are faster to train than LSTMs. Both of this variants of RNNs achieve similar results for the same tasks and can be used interchangeably. For some problems one might be better than the other, so it is advisable to try both and see if one is in fact superior.

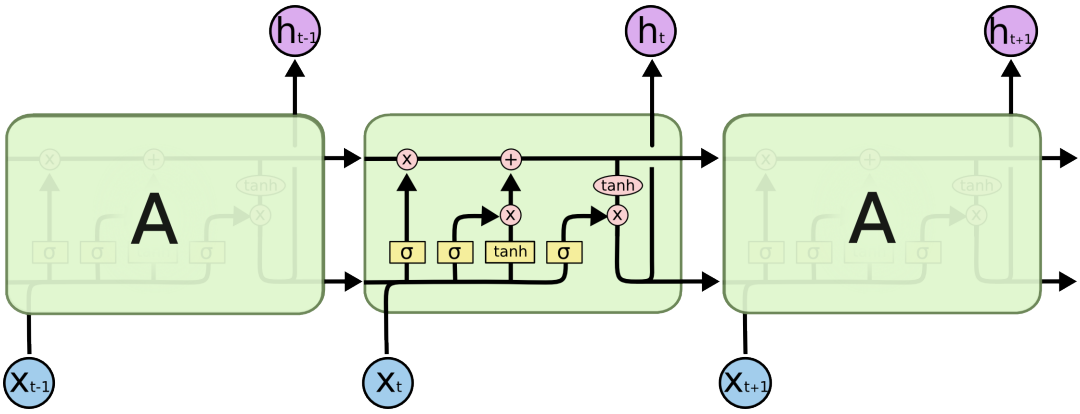

FIGURE 4.3: Architecture of LSTM

4.2 Convolutional neural networks - CNNs

Convolutional Neural Networks are mostly used in areas such as image recognition and classification but they are starting to become popular for solving problems in natural language processing as well.

Simply put, CNNs are layers of convolutions, mathematical combination of two functions that produce a third function, with nonlinear activation functions. In a traditional, feed-forward neural network, each input neuron is connected to each output neuron in the next layer. However in CNNs we do not use fully connected layers like in traditional NNs. Instead, the convolutions are used over the input layer to compute the output. The layers are connected locally and each part of the input is connected to a neuron in the output.

4.2.1 1D-CNN

CNNs share the same characteristics, the shape of the input data narrates how will the algorithm go through the data provided. To use CNNs on NLP tasks[1] they need to be one hot encoded and presented as a matrix, where each row is a vector that corresponds to one word or a character. When CNNs are used for image processing, the data is two-dimensional and two-dimensional kernels are used to slide over the data horizontally and vertically over the pixels, but when processing text we have only one-dimensional kernel that will slide horizontally over the matrix. All inputs should be the same size, so it is needed to define the maximum length of the input and add white space to the inputs that are smaller than the maximum length.

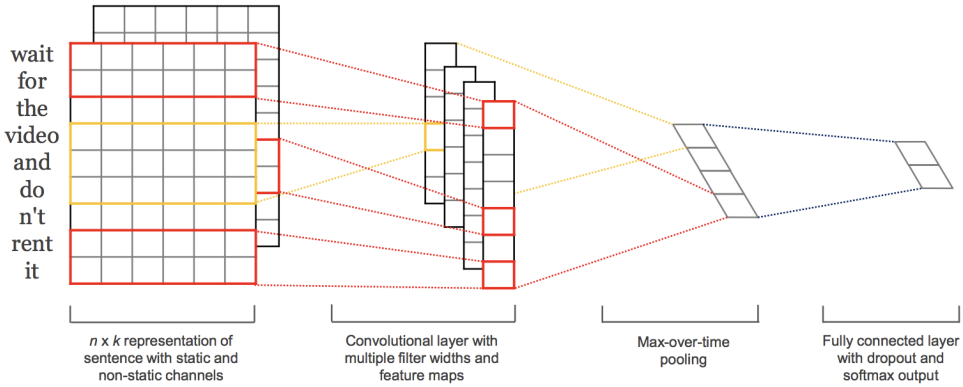

FIGURE 4.4: Scheme of 1D CNN

[1]Sentences or words are one-dimensional data as they follow each other sequentially to form a meaning.

Chapter 5

Autoencoders

Autoencoder (AE) is a type of unsupervised learning algorithm build with neural networks. The goal of an autoencoder is to learn an encoding for input data. It is composed of an encoder and a decoder. The task of the encoder is to take a high dimensional input and map it to a lower dimensional latent space and the job of the decoder is to generate the original input by trying to reconstruct it from the latent space representation. The size of the hidden representation will influence the reconstruction. The less dimensions it has, less information it will be able to extract. The autoencoder will prioritize the important features of the input data and the output data will only be an approximation of the input data.

The downside of a simple autoencoder is that they cannot be used as generative models and cannot produce new data samples. If we give an AE input that it has never seen before, the decoder will not be able to produce a reasonable output. Decoder of VAE, on the other hand, will be able to give us a reasonable output.

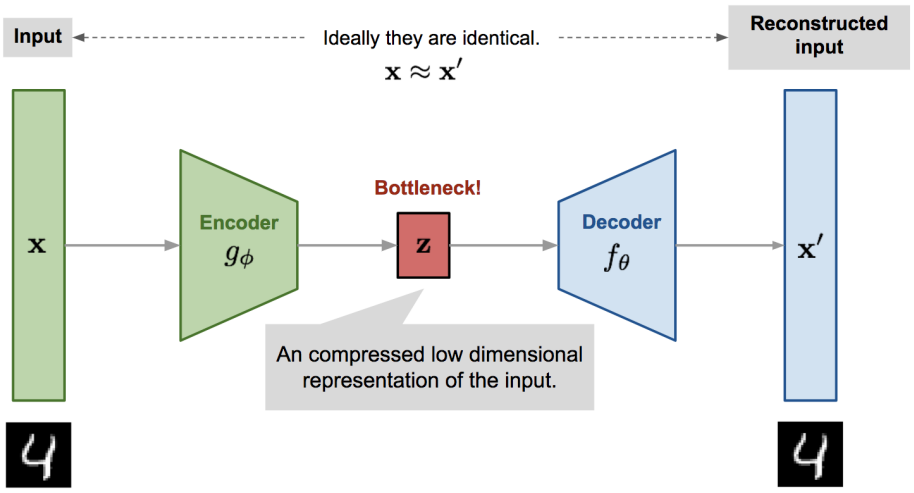

FIGURE 5.1: Autoencoder architecture

5.1 Variational Autoencoder

Variational autoencoders work as a generative model unlike plain autoencoders. The inputs are defined by using a standard normal distribution and the job of the decoder is to sample points from this distribution and produce a reasonable output. Encoder does not only create a single point in the latent space, but it creates a probability distribution. These latent distributions are Gaussian distributions of the same dimensional as the latent space.

The VAE objective:

$$E[logP(X|z)] - D_{KL}[Q(z|X)||P(z)] \tag{5.1}$$

where:

E: expectation value
P, Q: probability distributions
D_{KL}: Kullback-Leibler divergence
X: data
z: latent spaces

In autoencoders $Q(z|X)$ approximated by an encoder and $P(X|z)$ approximated by a decoder. New data x is generated from latent variables z. The goal is to have encoding distribution, $P(X|z)$, to be close to the data. But instead of using $P(X|z)$, approximated posterior $Q(z|X)$ is used.

As we can see the VAE objective, or the loss function of VAE, has two parts, the first part measures the quality of the autoencoding (the error between the original sample and its reconstruction) and the job of the second part, the Kullback-Leibler divergence, is to make the distribution of the encoder output as close as possible to a standard multivariate normal distribution.

5.2 Conditional Variational Autoencoder

The ultimate goal of molecular design for new materials and drugs is to directly generate molecules with the desired properties. Decoder in VAE cannot produce an output with specific properties on demand, but CVAE can. During the training time, the input data is being fed with its labels concatenated. The latent space does

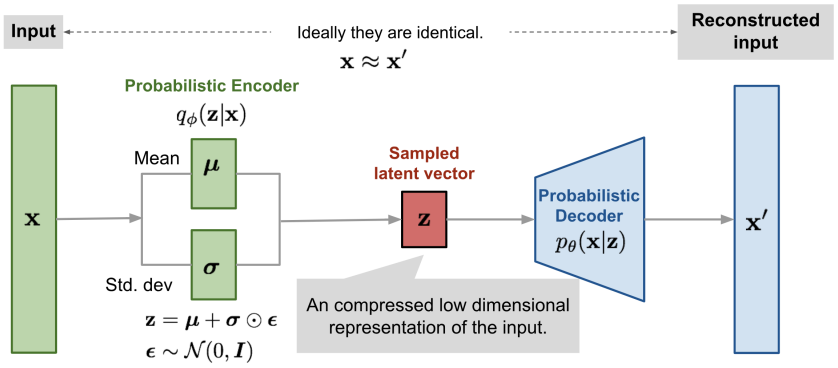

FIGURE 5.2: Architecture of VAE

not encode the data point, but it encodes information about the provided points, so CVAE allows us to generate the data with specific attribute.

To create CVAE from VAE, conditional vectors are simply concatenated into the existing neural net, to both encoder and decoder. A key difference of the CVAE from the VAE is to embed the conditional information in the objective function of the VAE, leading to the revised objective function as follow: where c denotes a condition vector. The condition vector c is directly involved in the encoding and decoding processes.

The CVAE objective:

$$E[logP(X|z,c)] - D_{KL}Q(z|X,c)||P(z|c)] \tag{5.2}$$

where:

E: expectation value
P, Q: probability distributions
D_{KL}: Kullback-Leibler divergence
X: data
z: latent spaces
c: condition vector

CVAE objective only differs from the VAE in term of the conditin vector c.

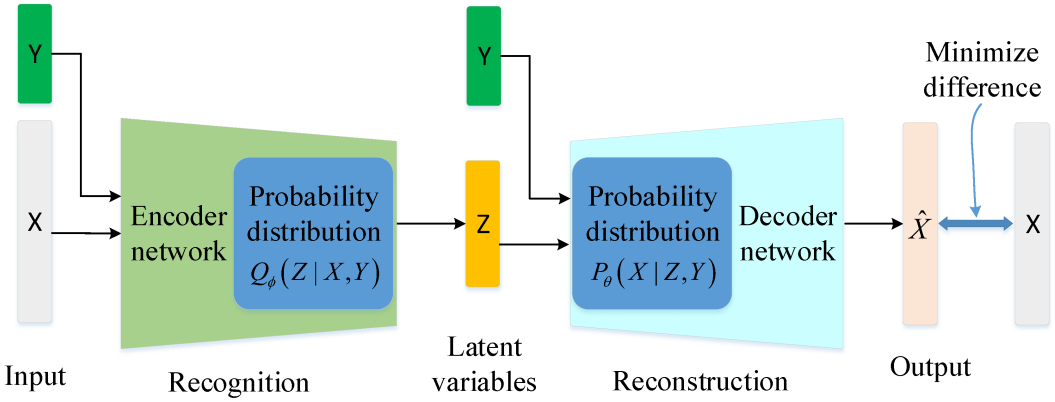

FIGURE 5.3: Architecture of CVAE - as seen the only difference between VAE and CVAE is the condition y.

5.3 Overview of Other Generative Models

5.3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) can learn to mimic any distribution of data. They consist of two neural networks, one called generator that generates new data and the discriminator that evaluates the generated images. The job of the discriminator is to determine if the outputs of the generator belong to the actual training data set or not, in other words, it is trying to recognize if they are authentic or fake.

As seen in Figure 5.4, generator is fed some data and it generates real looking fake images. Those images are then fed with the real images to the discriminator that then tries to classify them as real or not.

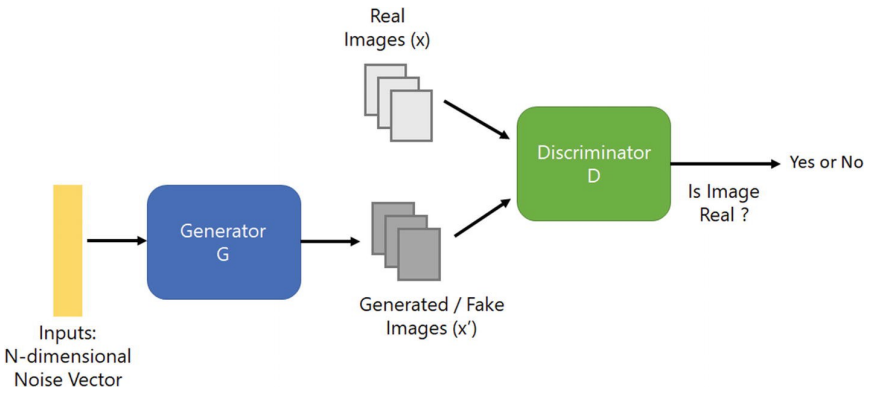

FIGURE 5.4: Illustration of Generative Adversarial Network architecture: The generator is creating new synthetic images, that should pass as authentic and gives them to the discriminator. The discriminator is trying to identify those images as fake.

5.3.2 Graph based Models

Graphs are fundamental data structures that capture the relational structure well. We would like to generate realistic graphs that are similar to the training data set and we would like to achieve graphs that optimize given constraints. Graph models can be used for generating drug like molecules either from an empty graph or add to an existing graph to achieve desired molecular properties.

Both graph generative models (Li, Zhang, and Liu, 2018; Li et al., 2018) mentioned in subsection 3.2.3 are a sequential graph generators. They start to build new compounds from an empty graph and then the generator creates one node at a time and connects each node to already generated nodes by creating edges.

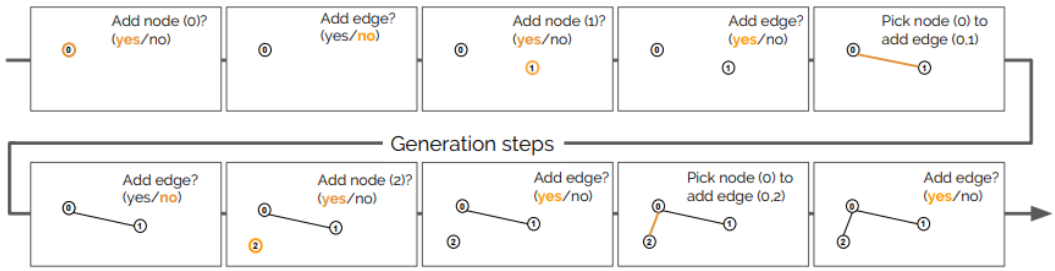

FIGURE 5.5: Depiction of the steps taken during the generation process (Li et al., 2018).

Chapter 6

VAE for Molecule Generation

6.1 Model Description

The Variational Autoencoder used was written by the team from ChEMBL (Blog, 2017) and is based on paper written by Gómez-Bombarelli et al. (2018). The model was trained on SMILES from ChEMBL 23[1] database. The team decided to only use SMILES that are no longer than 120 characters, same was done in the original paper. This excluded ∼3% of their data. 80% of the data was used for training and 20% was used for their test set, the model was trained with a validation accuracy of 0.99.

The authors of the original paper Gómez-Bombarelli et al. (2018) mention that they performed a random optimization over hyperparameters specifying the deep autoencoder architecture and training, such as the number of hidden layers, layer sizes, regularization and learning rates. They trained the model on different data sets with different latent space dimensions.

Their final model has the following architecture: encoding layer has three 1D convolutional layers, a flattening layer and a dense layer while the decoder consists of a dense layer, repeat vector and 3 layers of gated recurrent units. The ChEMBL team followed the architecture of the encoder and the decoder and set the number of dimensions in their latent space to 292. Architecture of the full model can be found in Appendix A.

Once the smiles shorter than 120 characters are padded with white space and one hot encoded they are fed into the encoder. The encoder encodes the input data and outputs parameters describing a distribution for each dimension in the latent space. In this case the latent space is 292-dimensional. The two output vectors are describing the mean and variance of the latent state distributions because it is assumed that prior follows a normal distribution. So, for a given SMILES, the encoder produces a distribution. A latent vector is then sampled and fed into the decoder

[1] Database of bioactive molecules with drug-like properties.

which decodes it into artificial SMILES.

6.2 Latent Space Visualization

I wanted to visualize the latent space of SMILES to see if the model will generate smooth and nicely populated. The latent space was visualized in two dimensions using principal component analysis. As we can see in Figure 6.1 by visualizing *sim*8000 SMILES from the DrugBank database, the latent space is nicely populated without any large empty areas.

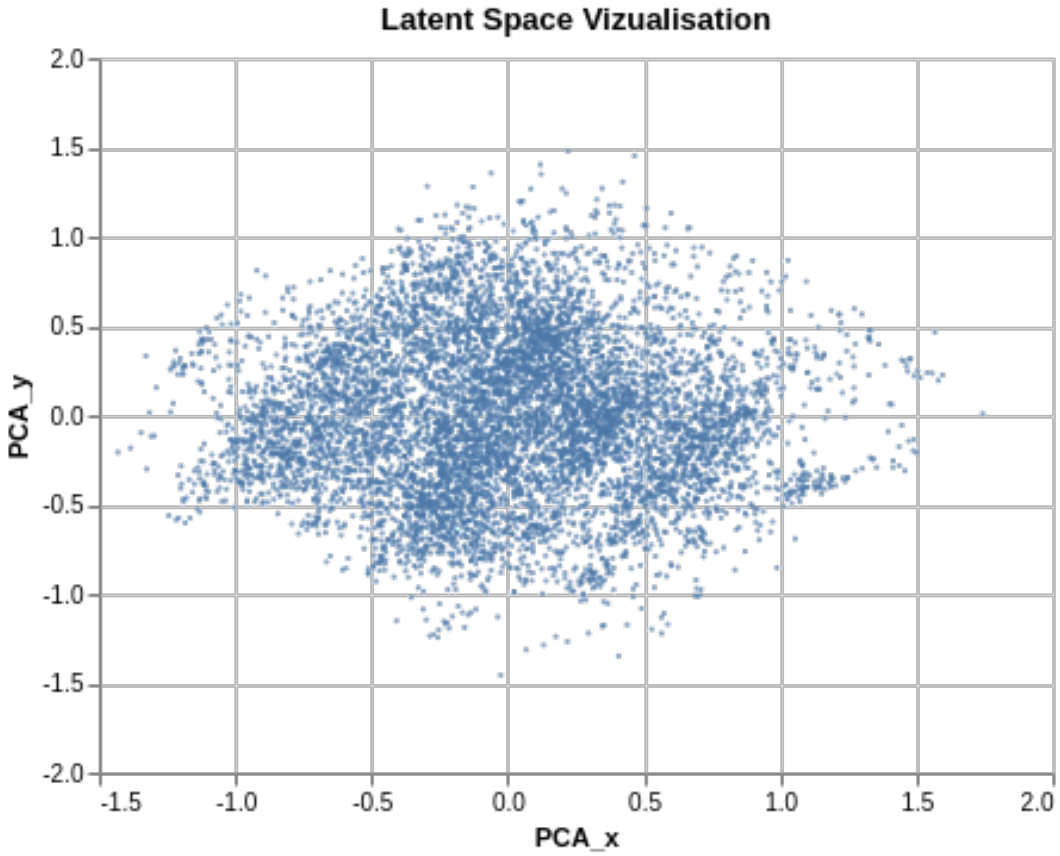

FIGURE 6.1: 2D vizualisation of ∼8000 molecules encoded into the latent space.

6.3 Experiments

For model evaluation, I looked at some specific examples of SMILES of different lengths and performed different experiments. Generally the longer the smile, the more complex the molecule. I visualized results using RDKit Python package.

Reconstruction of SMILES

First experiment was a simple reconstruction of a few different sized SMILES. SMILES were fed one by one into the model. The encoder created latent representation and the decoder tried to reconstruct the SMILES from that representation. I repeated the process 1000 times for each molecule. The model had no trouble reconstructing smaller, less complex molecules. It also occasionally changed an atom but those changes were usually very small. Example of such reconstruction can be seen in Figure 6.2. When I tried to repeat the process the more complex SMILES, there were only a few valid reconstructions and they were quite different from the original output.

Sampling from the noised Latent Space

The main goal is to generate molecules that are similar to the input molecule. To achieve that we can encode the molecule to its latent representation and add Gaussian noise. The results varied depending on the standard deviation chosen. After some experiments, I noticed that adding standard deviation around 0.05 produced wanted results. For smaller standard deviations the model only created a few unique smiles which were very similar to the original molecule or were original molecule and larger standard deviation produced molecules that were very different from the input one. Figure 6.3 shows results for standard deviation of 0.05. To see how close the valid SMILES are to the original, I visualized the latent space of all sampled points with PCA (see Figure 6.4).

Tanimoto Similarity

In Figure 6.5 we can see the difference between the similarity of molecules when using different standard deviation when sampling from noised latent space for Morgan Fingerprint radius of 2. When using standard deviation of 0.05, we can that the number of samples (frequency) is much larger and that there are many samples that are highly similarity to the input. On the other hand, when using standard deviation of 0.1 the number of produced samples is smaller and their structure is not very similar to the original.

In summary, standard deviation bigger than about 0.05 results in fewer valid samples that are not as closely related to the original molecule. Standard deviation of around 0.05 produced molecules closely related to the input molecules. Besides that, such standard deviation produced more samples. That kind of behaviour is expected. The higher the standard deviation the further we are away from the mean, the bigger difference in samples is expected to be seen.

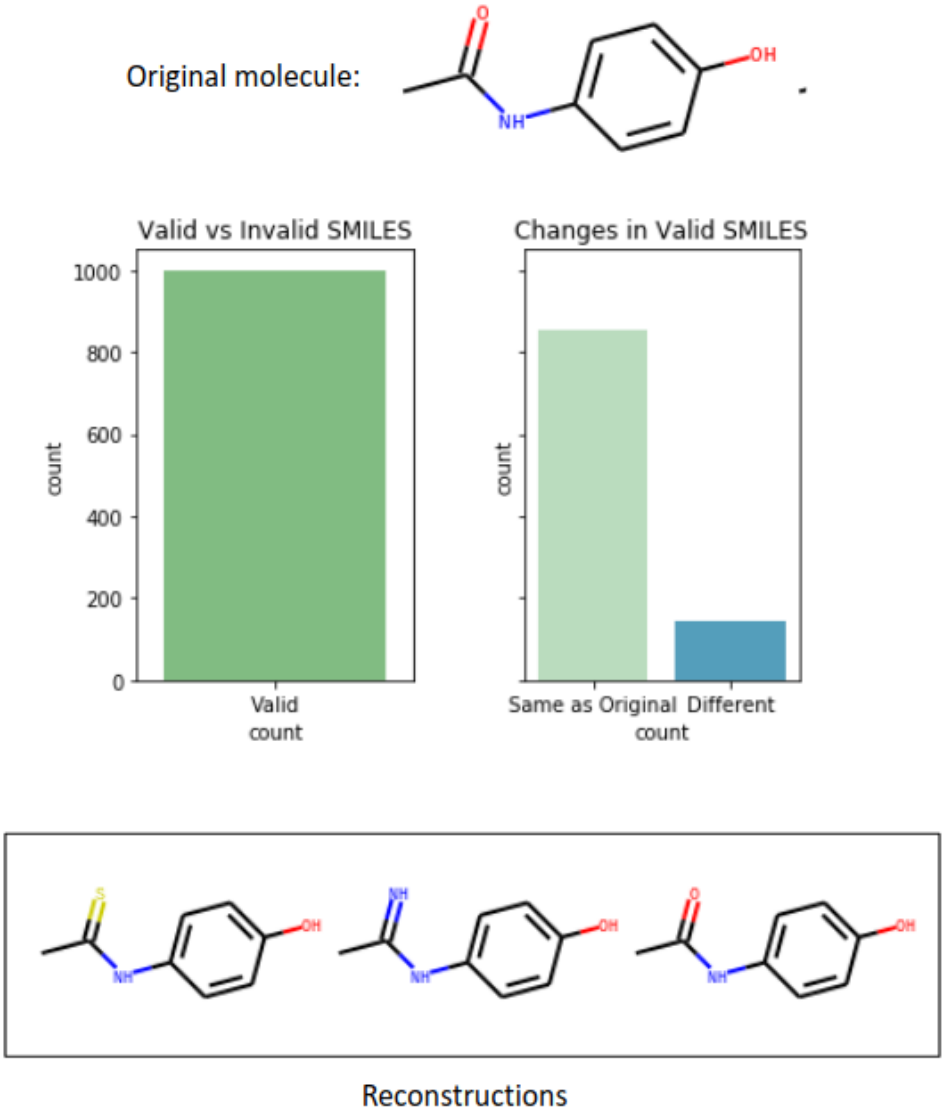

FIGURE 6.2: Left count plot is showing how many times the reconstruction of the input molecule was valid. The right count plot is showing how many time the reconstructed molecule was the same as original and how many times it was reconstructed differently. The bottom three molecules are all different reconstructions produced.

Interpolation

6.3.1 Interpolation

Some AEs (e.g. VAEs and CVAEs) are capable of learning smooth latent space representations of the input data and are being able to perform interpolations for real samples along any arbitrary axis. We need to choose a starting point and an end point and do the interpolation.

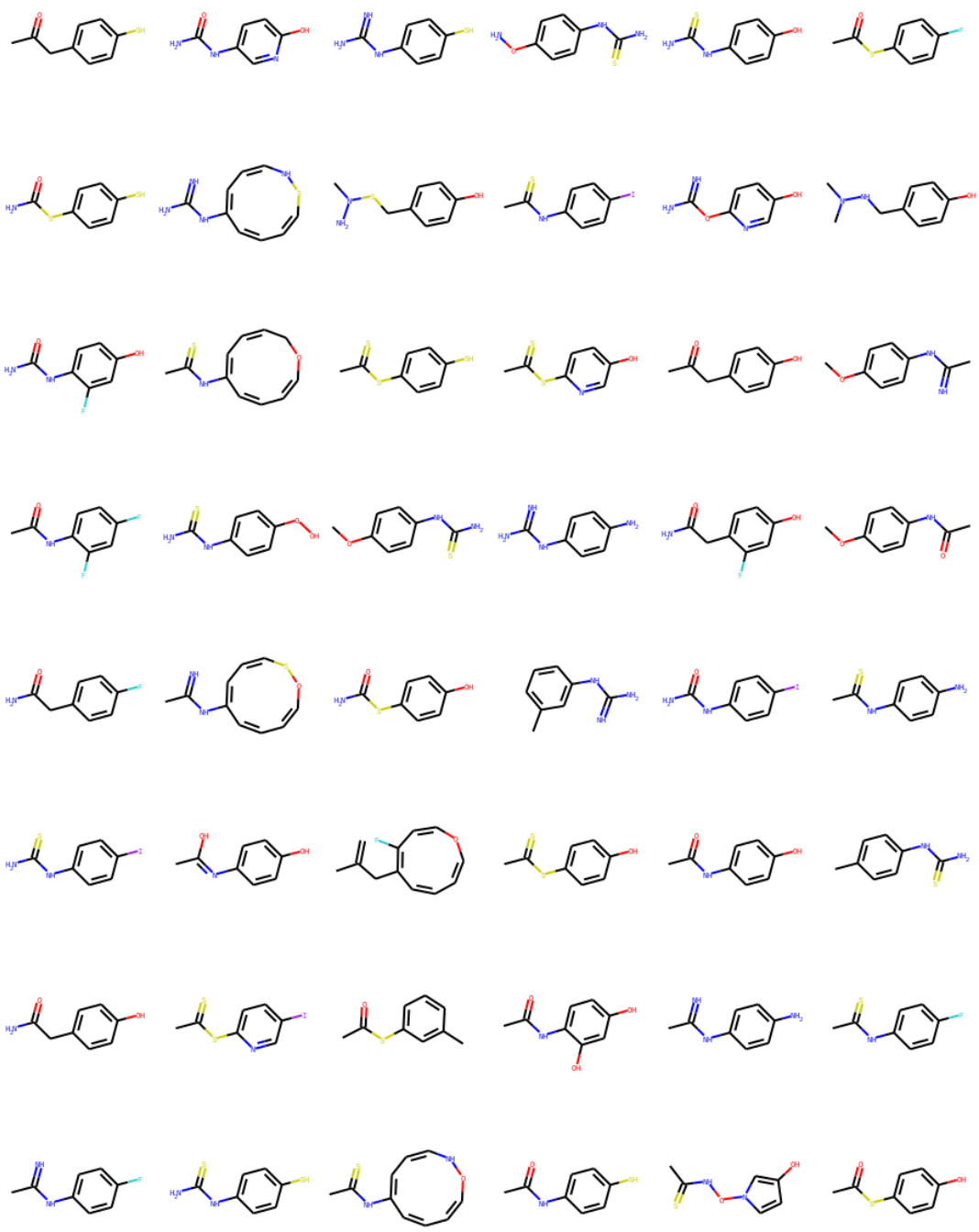

FIGURE 6.3: Examples of sampled molecules. std = 0.05

However we have to be careful in high dimensional spaces because the volume of the space is enormous and the data it contains is sparse. Interpolating between the two points linearly might go through an area of low probability, so spherical interpolation is needed.Path between two points being interpolated is a circular arc lying on the surface of a N-dimensional sphere.

Function for interpolation was already added to the code, but it was written for

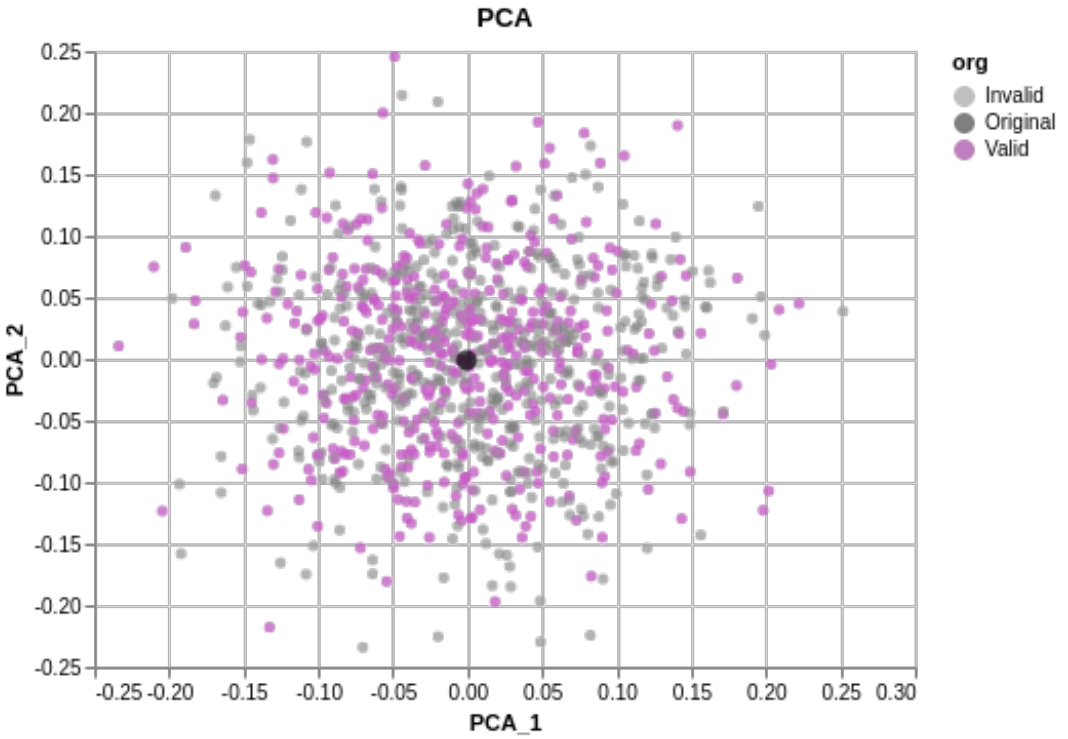

FIGURE 6.4: 2D visualisation of 1000 sampled data points using PCA.

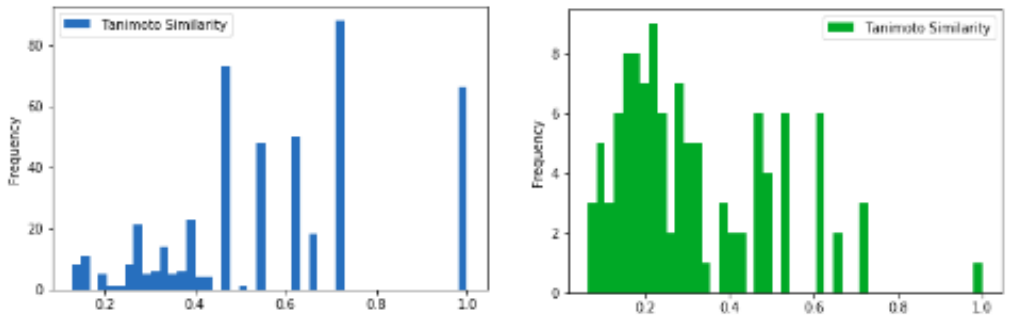

FIGURE 6.5: Tanimoto similarity done by comparing Morgan Fingerprints.

linear and not polar interpolation. Because the latent space has 292 dimensions we need to perform spherical interpolation. Figure 6.6 shows the results of spherical and linear interpolation between Aspirin and Paracetamol. Spherical interpolation produced twice as many results fro the same about of steps than linear interpolation. This is due to the fact that when interpolating linearly the path went through populated space.

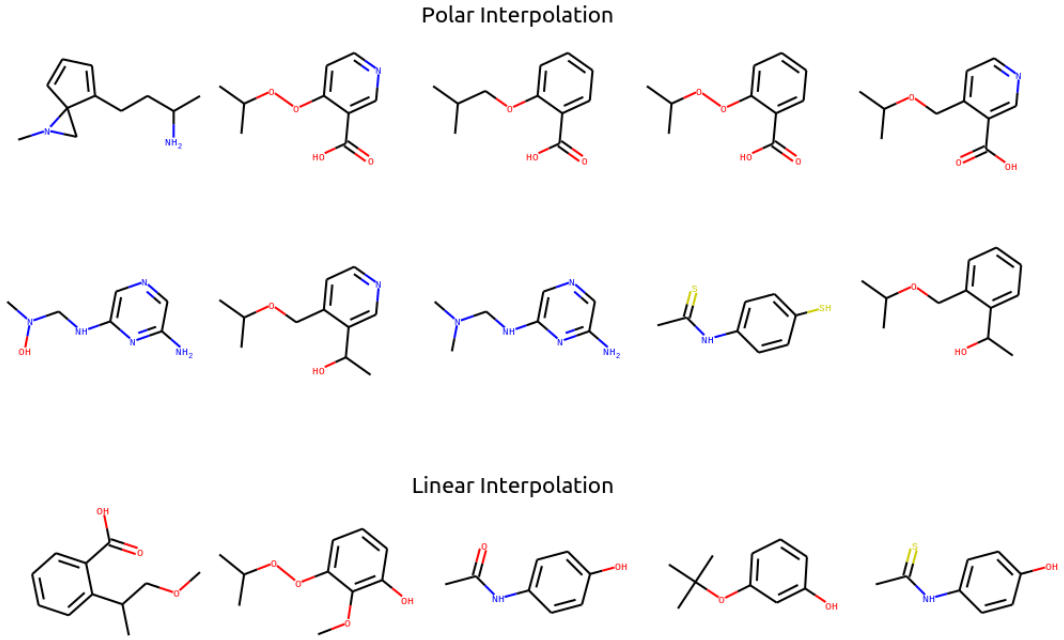

FIGURE 6.6: Polar vs. Linear interpolation

Chapter 7

CVAE for Molecule Generation

Although variational autoencoders are quite good at generating new samples, these samples do not necessary have the properties we want. Conditional variational autoencoders allow us to generate molecules with desired properties. The difference between the two is that CVAE needs an extra input that is going to denote these properties. For this project the properties are so called Signaturesduran2019extending. Signatures are vector format of bioactivity data. Each kind of signatures expressed different molecular properties. There are 25 different signatures and in this project we used signature B4 which stores information about binding. The model was trained using ∼400 000 samples.

In CVAE model, the molecular properties we want to control are represented as the condition vector. The CVAE model's training data is a composed of two parts SMILES and its corresponding signatures which are our condition vectors. Therefore, the desired molecular properties should be embedded in the final structure by determining a condition vector. The structure can be controlled separately from properties, unless the physical and chemical laws do not allow it. So, when we change the signature, we would like to see molecular structure change according to those.

7.1 Experiments

Reconstruction

CVAE seemed to perform better when reconstructing signatures. For the first experiment I encoded and decoded SMILES with the same signature and the valid SMILES were decoded as input molecule most of the time. More complex molecules were decoded with minor changes as well. Example for 'O=C(c1ccc(F)cc1)C1CCNCC1' SMILES can be seen in Figure **??**. Just like in the first part I encoded and decoded the molecule and the signature 1000 times.

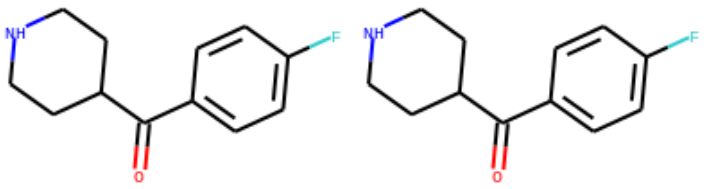

FIGURE 7.1: Original and decoded SMILES with original signature - O=C(c1ccc(F)cc1)C1CCNCC1.

For the second reconstruction I changed the SMILES's signature to see how that would change the outcome. As expected the structure of the input SMILES was changed. The rate of invalid SMILES was slightly lower, however, still quite high for more and less complex smiles. Figure 7.2 shows an example of the change seen in SMILES.

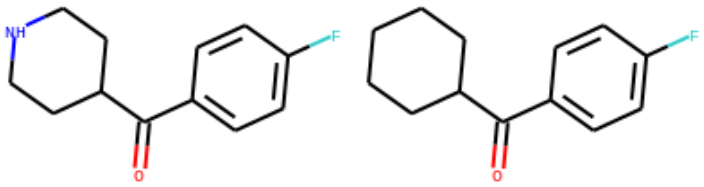

FIGURE 7.2: Original and decoded SMILES with changed signature - O=C(c1ccc(F)cc1)C1CCNCC1. The molecule on the left is the original molecule.

Sampling from Noised Latent Space

Just like with VAE, the standard deviation had large influence on generated molecules. I adopted standard deviation of 0.05 for the experiments. For the first example I used the original signature to compare with results when I changed it. As can be seen from Figures 7.3 and 7.4 that the sampled molecules are similar to the original one, but when comparing they look quite similar, even when looking at their Tanimoto plots.

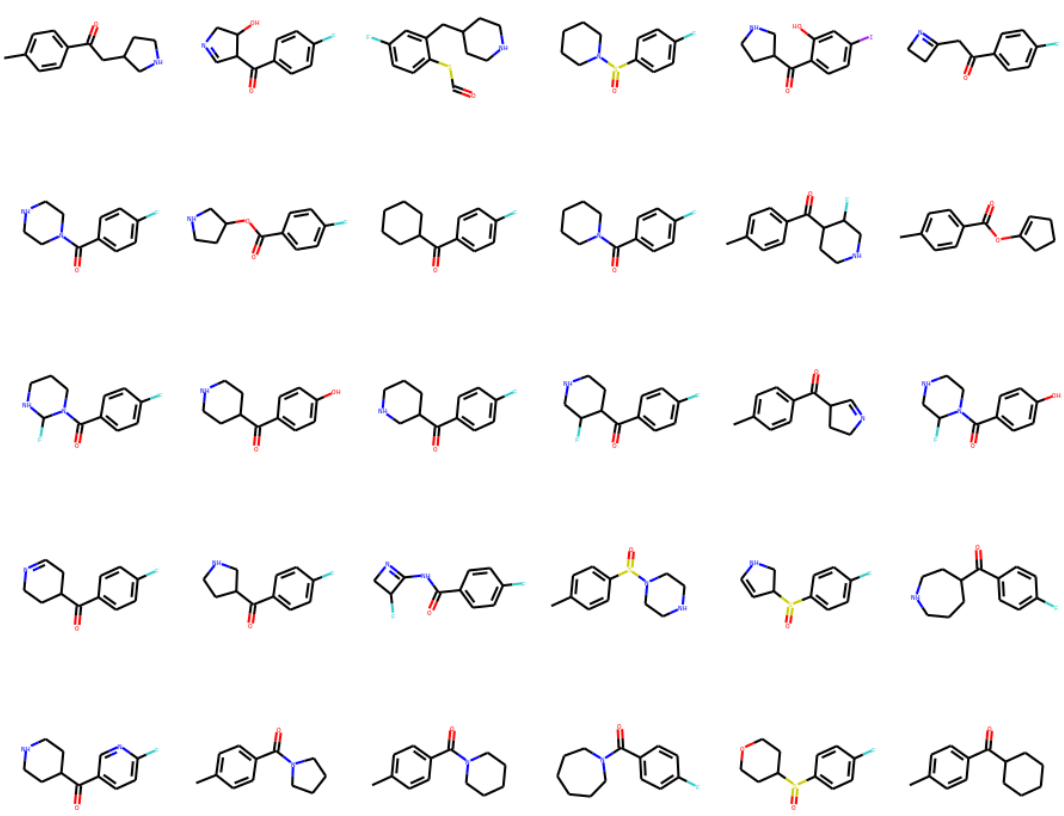

FIGURE 7.3: Sampled molecules using original signature and standard deviation of 0.05.

CVAE Conclusion

Because the CVAE is meant for building molecules with desired properties, interpolation does not seem necessary as molecules can be redesigned by changing their signatures.

in conclusion CVAE was able to reconstruct original molecules even when we change their signatures, which is its main purpose. Because signatures are quite complex it was hard to come to any solid conclusions when changing them when looking at specific molecules. It is safe to say that it works as the decoded molecules have different structures. To completely conclude if the CVAE is useful, we have to perform more detailed analysis.

I think that in the case of CVAE I should also try comparing the signatures and not the molecular structures to better understand the changes when the condition is adjusted.

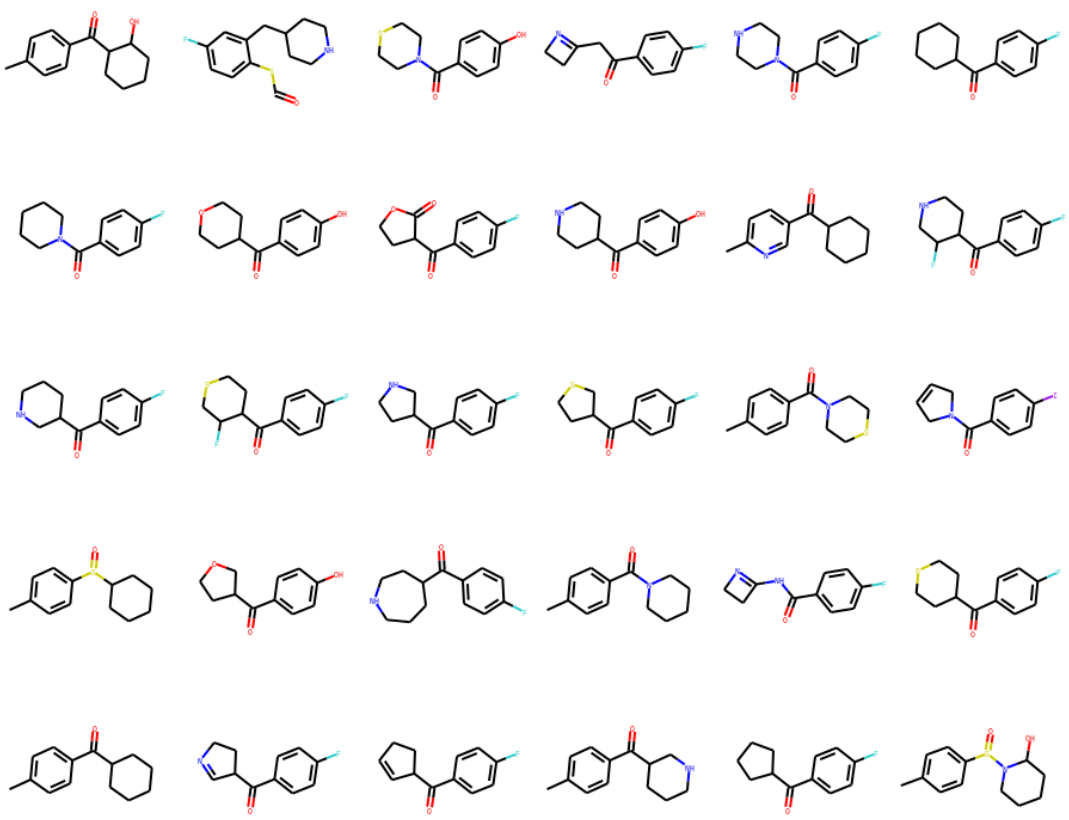

FIGURE 7.4: Sampled molecules using different signature and standard deviation of 0.05.

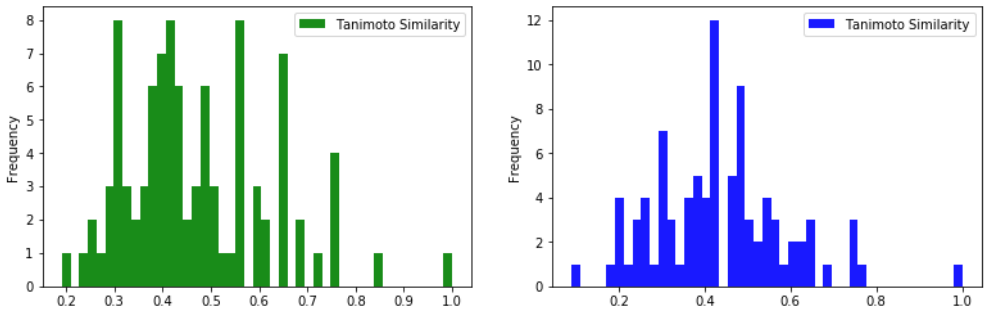

FIGURE 7.5: Tanimoto distances for molecules sampled with the original signature on the left and changed signature on the right.

Chapter 8

Discussion and Conclusion

Generative models have been emerging in the field of drug design. For this project we looked at two generative models: variationa autoencoder and conditional variational autoencoder.

While it was easy to see where VAE had problems, CVAE was much harder to evaluate the same way. Both of the models seemed to work, but to really know how well the VAE and CVAE performed at generating novel compounds, further analysis of results, such analysis includes calculation of drug likeness, synthetic accessibility score and physical properties, should be done.

Even though they both worked fairly well there is a lot that can be done to improve them. Both of had to learn grammar rules of the SMILES, so adding the rules while training would improve their Reconstruction results. According to the literature, the best choice for such task would be graph based models and generative adversarial networks. These models are much better at generating synthetic data than autoencoders but are much harder and much more expensive to train.

Appendix A

Model Architecture

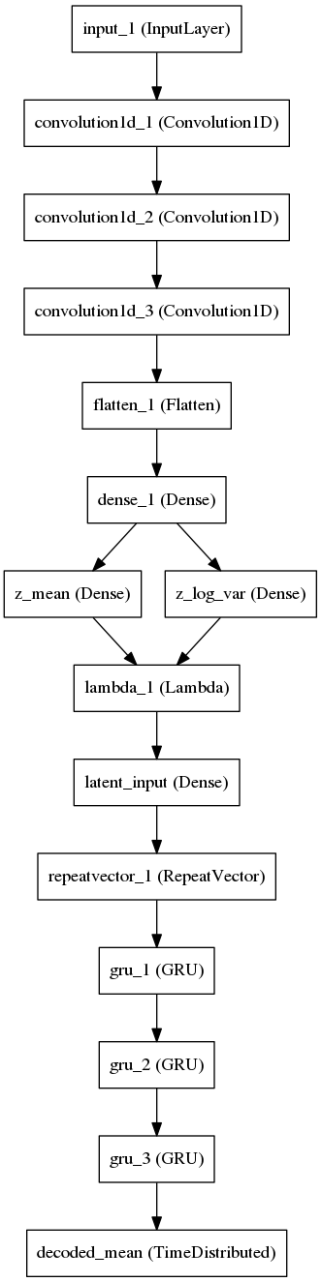

FIGURE A.1: VAE Architecture

Appendix B

Appendix

B.1 GitHub Repository

https://github.com/spela-z/Thesis

B.2 Bibliography

General

- "Semi-supervised learning with deep generative models", Kingma et al. (2014)
- "The rise of deep learning in drug discovery", Chen et al. (2018)
- "Tutorial on variational autoencoders", Doersch (2016).
- "Inverse molecular design using machine learning: Generative models for matter engineering", Sanchez-Lengeling and Aspuru-Guzik (2018)

References

Bajusz, Dávid, Anita Rácz, and Károly Héberger (2015). "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?" In: *Journal of cheminformatics* 7.1, p. 20.

Bjerrum, Esben and Boris Sattarov (2018). "Improving Chemical Autoencoder Latent Space and Molecular De Novo Generation Diversity with Heteroencoders". In: *Biomolecules* 8.4, p. 131.

Bjerrum, Esben Jannik and Richard Threlfall (2017). "Molecular generation with recurrent neural networks (RNNs)". In: *arXiv preprint arXiv:1705.04612*.

Blaschke, Thomas et al. (2018). "Application of generative autoencoder in de novo molecular design". In: *Molecular informatics* 37.1-2, p. 1700123.

Blog, ChEMBL (2017). *Using Autoencoders for Molecule Generation*. URL: https://github.com/chembl/autoencoder_ipython (visited on 08/05/2019).

Chen, Hongming et al. (2018). "The rise of deep learning in drug discovery". In: *Drug discovery today* 23.6, pp. 1241–1250.

Cho, Kyunghyun et al. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078*.

De Cao, Nicola and Thomas Kipf (2018). "MolGAN: An implicit generative model for small molecular graphs". In: *arXiv preprint arXiv:1805.11973*.

Doersch, Carl (2016). "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908*.

Gómez-Bombarelli, Rafael et al. (2018). "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2, pp. 268–276.

Gupta, Anvita et al. (2018). "Generative recurrent networks for de novo drug design". In: *Molecular informatics* 37.1-2, p. 1700111.

Hartenfeller, Markus and Gisbert Schneider (2011). "Enabling future drug discovery by de novo design". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1.5, pp. 742–759.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Kadurin, Artur et al. (2017). "druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico". In: *Molecular pharmaceutics* 14.9, pp. 3098–3104.

Kim, Sunghwan et al. (2015). "PubChem substance and compound databases". In: *Nucleic acids research* 44.D1, pp. D1202–D1213.

Kingma, Durk P et al. (2014). "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems*, pp. 3581–3589.

Kusner, Matt J, Brooks Paige, and José Miguel Hernández-Lobato (2017). "Grammar variational autoencoder". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1945–1954.

Li, Yibo, Liangren Zhang, and Zhenming Liu (2018). "Multi-objective de novo drug design with conditional graph generative model". In: *Journal of cheminformatics* 10.1, p. 33.

Li, Yujia et al. (2018). "Learning deep generative models of graphs". In: *arXiv preprint arXiv:1803.03324*.

Lim, Jaechang et al. (2018). "Molecular generative model based on conditional variational autoencoder for de novo molecular design". In: *Journal of cheminformatics* 10.1, p. 31.

Lionta, Evanthia et al. (2014). "Structure-based virtual screening for drug discovery: principles, applications and recent advances". In: *Current topics in medicinal chemistry* 14.16, pp. 1923–1938.

McInnes, Campbell (2007). "Virtual screening strategies in drug discovery". In: *Current opinion in chemical biology* 11.5, pp. 494–502.

Newman, David J and Gordon M Cragg (2016). "Natural products as sources of new drugs from 1981 to 2014". In: *Journal of natural products* 79.3, pp. 629–661.

Rupakheti, Chetan et al. (2015). "Strategy to discover diverse optimal molecules in the small molecule universe". In: *Journal of chemical information and modeling* 55.3, pp. 529–537.

Sanchez-Lengeling, Benjamin and Alán Aspuru-Guzik (2018). "Inverse molecular design using machine learning: Generative models for matter engineering". In: *Science* 361.6400, pp. 360–365.

Schneider, Gisbert (2018). "Automating drug discovery". In: *Nature Reviews Drug Discovery* 17.2, p. 97.

Segler, Marwin HS et al. (2017). "Generating focused molecule libraries for drug discovery with recurrent neural networks". In: *ACS central science* 4.1, pp. 120–131.

Simonovsky, Martin and Nikos Komodakis (2018). "Graphvae: Towards generation of small graphs using variational autoencoders". In: *International Conference on Artificial Neural Networks*. Springer, pp. 412–422.

Virshup, Aaron M et al. (2013). "Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds". In: *Journal of the American Chemical Society* 135.19, pp. 7296–7303.