Deep Generative Modelling in Systems Medicine: From Transcriptomics Data to Drug Development

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Marie Oestreich

aus

Goch

Bonn, Juni 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Joachim L. Schultze

2. Gutachter: Prof. Dr. Mario Fritz

Tag der Promotion: 15.12.2023

Erscheinungsjahr: 2024

Abstract

Artificial intelligence (AI) has been widely applied across many domains, including the biomedical research field. These AI models mostly include discriminative models such as classifiers and regressors that predict for example diseases, disease progression or susceptibility to drugs based on patient data. Outside of the medical field, however, there has been a recent surge of another type of AI: generative models that allow to create data rather than discriminate it. In this work, I illustrate the opportunities offered by generative models in the context of systems medicine on two application scenarios.

Firstly, I investigate the utility of several types of generative models on the task of generating synthetic transcriptomics data. To unlock the full potential of transcriptomics data, making it widely available to other researchers is essential. However, its highly personal nature requires measures to protect patient privacy. In this work, I investigate generative AI for the generation of private synthetic patient cohorts with the goal to enable sharing of transcriptomics data without privacy-violations. I evaluate the synthetic cohorts generated by models that do or do not protect patient privacy for biological soundness. This includes the data's utility to train classifiers for real data, as well as the preservation of differential expression and co-expression of genes.

Secondly, I address generative modelling for the property-based *de novo* design of compounds. I firstly focus on the problem of AI-learnable representations of molecules and particularly consider the requirements of such a representation and investigate how well they preserve chemical information content. Specifically, I research the impact of the model architecture of autoencoders on the embedding quality of encoded molecules. From that, I devise an optimisation strategy that not only enhances embedding quality, but also reduces the resources required for training the autoencoder. In a time where global warming progresses at an alarming speed and where resources are waning, measuring the utility of new inventions needs to include the consideration of its environmental impact. I then consider the task of devising new compounds with generative AI. Here, my focus lies especially on the guided generation based on desired properties. During the model design, I additionally concentrate on devising an architecture that is flexible, allowing easy addition of further properties without retraining the model. I demonstrate the effectiveness of the approach for single-property as well as multi-property conditioning.

Zusammenfassung

Künstliche Intelligenz (KI) findet mittlerweile in vielen Bereichen Anwendung, darunter auch in der biomedizinischen Wissenschaft. Bei diesen KI-Modellen handelt es sich größtenteils um diskriminative Modelle, wie zum Beispiel Klassifikatoren oder Regressoren, die beispielsweise für die Prognose von Krankheiten, Krankheitsverläufen oder der Empfänglichkeit gegenüber Medikamenten genutzt werden. Außerhalb der Medizin wendet sich die Aufmerksamkeit jedoch zunehmend einem anderen Typen von KI zu: Den generativen Modellen, die es ermöglichen, neue Datenpunkte zu generieren, statt existierende Daten zu klassifizieren. In dieser Arbeit illustriere ich anhand von zwei Anwedungsbeispielen die Möglichkeiten, die generative Modelle im Bereich System Medizin bieten.

Zuerst untersuche ich die Nützlichkeit von verschiedenen Arten von generativen Modellen in Bezug auf die Generierung synthetischer Transkriptomdaten. Um das volle Potential von Transkriptomdaten auszuschöpfen, ist es von essentieller Bedeutung, die Daten anderen Wissenschaftlern frei zur Verfügung zu stellen. Jedoch sind diese Daten hoch sensitiv und erfordern daher Maßnahmen, die die Privatsphäre der Patienten schützen. In dieser Arbeit untersuche ich generative künstliche Intelligenz für die Generierung Privatsphäre schützender künstlicher Patienten-Kohorten mit dem Ziel, diese offen zu teilen, ohne die Privatsphäre der Patienten zu verletzen. Ich evaluiere die synthetischen Kohorten, die von Modellen generiert wurden, welche die Privatsphäre von Patienten schützen und von solchen, die dies nicht garantieren, auf biologische Robustheit. Dies berücksichtigt zum einen, wie geeignet die synthetischen Daten sind, um einen akkuraten Klassifikator für die echten Daten zu trainieren, und zum anderen auch die Erhaltung von differentiell exprimierten und ko-exprimierten Genen.

Anschließend adressiere ich die Anwendung generativer KI-Modelle für die eigenschaftsbasierte Entwicklung bisher unbekannter chemischer Substanzen. Dabei befasse ich mich zunächst mit dem Problem, Moleküle so darzustellen, dass sie für KI-Modelle lernbar sind. Ich betrachte insbesondere die Anforderungen, die eine solche Repräsentation erfüllen muss und untersuche die Erhaltung chemischer Informationen. Hier liegt der Fokus insbesondere darauf, wie die Modellarchitektur von Autoencodern die Kodierungsqualität der Moleküle beeinflusst. Darauf aufbauend leite ich eine Optimierungsstrategie ab, die nicht nur die Kodierungsqualität erhöht, sondern auch die Ressourcen für den Trainingsprozess des

Autoencoders reduziert. In einer Zeit, in der die globale Erderwärmung alarmierend schnell voranschreitet und Ressourcen schwinden, muss die Einschätzung der Nützlichkeit neuer Entwicklungen auch immer deren Einfluss auf die Umwelt mit einbeziehen. Anschließend untersuche ich die Generierung neuer Moleküle mit Hilfe generativer Modelle. Hier liegt der Fokus insbesondere auf der gesteuerten Generierung basierend auf gewünschten Eigenschaften, die das generierte Molekül erfüllen soll. Während der Modellentwicklung konzentriere ich mich außerdem auf eine flexible Architektur, die es ermöglicht, weitere Eigenschaften in den Generierungsprozess einzubinden, ohne das Modell erneut trainieren zu müssen. Ich demonstriere die Effizienz unseres Ansatzes in verschiedenen Szenarien, wobei die Generierung sowohl durch einzelne als auch durch mehrere Eigenschaften gleichzeitig gesteuert wird.

Table of Contents

List of Figures

List of Tables

1. Introduction

In this work, I will address the application of deep generative modelling in the context of systems medicine, with a particular focus on transcriptomics and drug-development. Systems medicine is a field of study that has emerged rather recently [1,2]. It strives to understand the systemic realisation of diseases and other stressors in the body using a wide variety of systemic data. Systems medicine has developed out of systems biology, inspired by its combination of cross-disciplinary efforts, systemic assessment of pathways and mathematical and computational methods for modelling and analysing data [1]. It expands upon systems biology with a strong focus on translation of basic research into clinical settings as well as data integration methods to combine and extract information from varying data modalities. This data in part originates from Omics technologies [2]. Omics technologies are high-throughput methods to capture large-scale data of a specific type of biomolecule from a biological sample [3], such as deoxyribonucleic acid (DNA) in Genomics [4], ribonucleic acid (RNA) in Transcriptomics [5], proteins in Proteomics [5,6], lipids in Lipidomics [7] or epigenetic modifications in Epigenomics [8]. The analysis of such datasets helps to understand the systemic implications of the studied conditions and provides a holistic entrypoint to developing treatments [9–14].

The work presented here will particularly address two application cases of generative modelling in systems medicine, illustrated in three chapters (Figure 1): Firstly, in the ProGeneGen project, I will address the acute issue of sharing high-dimensional medical data without violating patient privacy exemplified on transcriptomics data. Sharing transcriptomes acquired over the course of a study is essential, not only in multi-party studies but also to make it available to researchers that are not involved in the study but that might benefit from the information held in the data [15–17]. However, these datasets are highly privacy sensitive since they contain concise quantitative measurements of the expression levels of genes in an individual [16–18]. Hence, the goal of the ProGeneGen project is the development and assessment of privately trained generative models to generate privacy-preserving synthetic data that can be shared instead of the original without risking privacy violations of the patients.

Secondly, I will demonstrate the potential of generative modelling for the development of new chemical compounds with desired properties. Here, the first step will be to transform the molecular data into a representation that is machine-readable and learnable, particularly such that they can be handled by machine learning models consisting of deep neural networks. A model type referred to as autoencoders (AE) is frequently applied to this task [19–21], however there is a lack of consensus in the used architectures and little understanding of how

they influence the quality of the retrieved molecular encodings, particularly with respect to how well they represent chemical similarity. Therefore, the second project presented here is a systematic evaluation of autoencoder architectures, their impact on encoding behaviour and how they can be optimised to boost performance. I additionally investigate how this optimisation can be conducted while simultaneously reducing the resources required for training. This is motivated by the concerning trend of AI exclusivity to groups of people with high resource-access as well as the devastating effect that training and deploying AI models have on the environment, founded in the increasing consumption of energy and other resources [22–24]. In a second step, I will then introduce the use of generative models, specifically diffusion models [25–28], to generate novel drug-like molecules with generative AI methods. Drug development is a laborious and costly process that does *not* yield an approved and market-ready product in 86.2% - 90,4% of the cases after the drug entered the phase 1 trial [29]. This is often due to the investigated drug candidates not having desired properties which negatively affects efficacy [30,31]. To improve the development process, AI models have been used to traverse the space of drug-like molecules faster and to narrow it down to a subset of promising candidates [32–34]. In this context, there is a particular demand for models that are capable of *de novo* generation of molecules with multiple target properties. Ideally, these properties do not only include physico-chemical properties but also complex systemic properties such as high binding-affinity to a target protein and drug-induced gene-expression profiles. The systemic nature of the gene expression data offers insights into complex, interdependent, high-level alterations under a given condition. It can thus serve as a functional assessment of the operating mode of a drug [35–37] and is utilised in the drug development process [38–41].

In summary, this thesis contributes scientific insights on three topics: 1) the private sharing of transcriptomic data to facilitate privacy-compliant research using generative modelling, 2) the impact of model architectures on the encoding quality of molecules and their optimisation for use in downstream machine learning tasks such as drug development while minimising the resource consumption of the training process and 3) generating novel drug-like molecules given multiple target properties as well as complex properties from systems medicine using diffusion models.

The following sections will introduce the required background information on concepts used throughout the three projects. These include an introduction to deep generative modelling with a particular focus on the model types used in this work, an overview on transcriptomic

data and an illustration of the need for privacy preserving sharing of transcriptomics datasets as well as the involvement of machine learning in drug development.

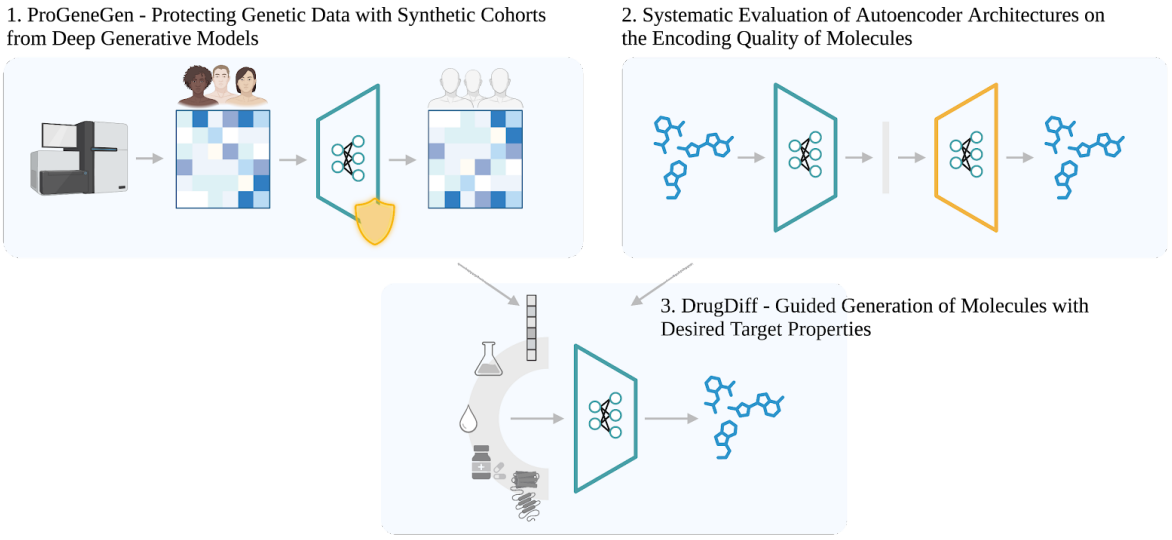


Figure 1: Overview of projects and their connection. The first project addresses the use of generative AI to create private synthetic cohorts of transcriptomic data that are freely shareable without violating patient privacy. The second project focuses on the impact architectural changes of autoencoders have on their ability to encode molecules with high reconstruction rate and a chemically meaningful latent space. It will further address how the architecture can be optimised to achieve good performance while simultaneously minimising resources. The third project is a generative model that uses latent representations of molecules for *de novo* generation based on desired target properties, including induced gene-expression. It thus utilises insights from the other two projects.

1.1 Deep Generative Modelling

Machine Learning models can be generally divided into two groups: discriminative models and generative models [42]. The group of discriminative models comprises classifiers as well as regressors and they generally strive to learn decision boundaries from a set of training data. They then utilise these boundaries to predict the label of an unseen datapoint. Such labels can be categorical, in the case of classification tasks, or numerical, in the case of regression models.

In contrast to discriminative models, generative models aim to use the distribution of a real-life data set and, based on that, generate new data points. If the model was trained successfully, then these data points are indistinguishable from the original data distribution. In the simplest case, such generative models can be purely statistics based and not require any machine learning at all. As an example, consider a generative model that generates dice rolls. This can be explicitly formulated by restricting the set of possible values to

$X = \{1, 2, 3, 4, 5, 6\}$ and assigning each value the probability of $p(x) = \frac{1}{6}$, $x \in X$. While this would generate realistic data points, it requires full prior knowledge of the underlying statistics. Since that is often not given in more complex, real-life data sets, *deep* generative models are used. These machine learning models aim to infer the statistics from a training set, therefore not depending on prior knowledge [43]. This benefit comes at the cost of requiring large training sets, where the size increases further the higher the complexity of the data is, in order to avoid the curse of dimensionality [44]. The model type used to build deep generative models is - as the name suggests - a deep neural network, which will be explained in more detail in the section below.

To assess the quality of a set of generated, i.e. synthetic, data the following domain- and model-agnostic metrics have been proposed [45]: 1) Fidelity, which quantifies the quality of the generated samples, 2) diversity, which assesses how well the generated samples cover the distribution of the original data, and 3) generalisation, which measures how innovative the model is and how well it avoids to copy data points from the original set.

A large variety of deep generative models exists and new ones are developed at a high pace [43]. Three model types are of particular interest to the work presented in this thesis: Variational autoencoders (VAEs) [46], generative adversarial networks (GANs) [47] and diffusion models [25–28]. The underlying concepts of these model types are described after an introduction to their building blocks: deep neural networks.

1.1.1 Deep Neural Networks

As previously mentioned, many real-life datasets are highly complex [44,48] and the rules that allow to make a prediction, to provide a label or to create a new data instance are often not known and can therefore not be explicitly formulated. In order to learn how to map an output to a set of input features provided by a data point in such complex cases, artificial neural networks are used [49,50]. These networks devise the rules for mapping outputs to inputs by observing a large set of training data. They learn by example instead of relying on prior knowledge.

The idea of artificial neural networks was inspired by the experience-based human learning-process. To mimic this learning process, the architecture of artificial neural networks is influenced by the brain [51,52]. The most basic unit of a neural network is a neuron. A collection of neurons forms the next higher structural entity - a layer. An artificial neural network needs to have at least one *hidden* layer - a layer that is neither the input nor the

output layer - to be considered a *deep* neural network [53]. Each neuron in a layer receives an input value v, multiplies this value with a weight w that connects the neuron to a neuron in the next layer, and finally adds a bias value b before passing the modified input onwards to the next layer [54] (Figure 2). Thus, the information flow passing through one single neuron can be described as

$$out = v * w + b.$$

The above formulation can be aggregated to describe the input-output-migration from one layer to the next as

$$v_i = v_{i-1} * W_i + b_i,$$

where v_{i-1} is the output vector of the previous layer, W_i is the weight matrix of the current layer and b_i is the bias vector of the current layer. Consequently, v_i is the resulting output of the current layer.

However, up to this point the described process is purely linear, which means that a series of layers could be condensed into one single function based on composition, collapsing the network to a single layer [55]. The strong suit of neural networks, though, is that they are also able to model non-linear relationships. Therefore, another piece is required to prohibit collapsing the network into one linear function: non-linearity. Non-linearity is introduced after hidden layers in the form of an activation function. Different activation functions exist, a popular one is the Rectified Linear Unit (ReLU), where $ReLU(x) = max(0, x)$, effectively setting all negative outputs to zero before passing them on to the next layer [55].

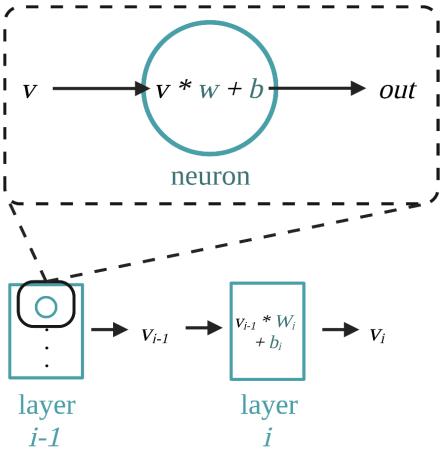


Figure 2: Basic operation mode of an artificial neural network. Input *v* is processed by each neuron (circle) in a layer (rectangle) by multiplying with a weight *w* and adding a bias *b*. Aggregated information from all neurons in a layer is passed on to the neurons of the next layer.

During the training process, the neural network is given feedback for its performance based on a loss function. A variety of loss functions exists, tailored for different tasks and output types. The loss function receives the final output of the neural network, the *prediction*, as well as the desired output, the *target* [56]. These can be labels or numeric values depending on the context. It then evaluates the network's performance based on how large the loss is: a small loss indicates that the model's prediction is very close to the target, while a large loss implies that the network's prediction was far off. To correct the network's mistakes, the weights are adjusted using a process called backpropagation [57]. The network is trained in an optimization process that aims to minimise the loss.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network that will play an important role in the second project presented in this work. I will therefore describe their history and the way they work in more detail.

Tracking down the origins of RNNs actually turned out trickier than expected! There does not seem to be *the* original RNN paper. Among the first are publications by Micheal I. Jordan in 1986 [58] and - building up on that - a publication by Jeffrey L. Elman in 1990 [59] that explicitly illustrate networks with recurrent connections. However, so-called *cyclic networks* were already mentioned in 1943 [51]. The general idea is to provide the network with a sort of memory that allows it to make predictions based on past experiences [60]. Like for neural networks in general, the inspiration came from the human mind: We are capable of including previously acquired knowledge into our decisions. For example, when we speak, we can remember how we started our sentence in order to finish it in a meaningful and coherent way. Many - if not most - of our actions are causal consequences of prior experiences. Many problems that we encounter *require* this ability to remember the past in order to be solved. To model such problems, for example generating and understanding speech [61], analysing video material [62–64], translating between languages [65], predicting weather [66] etc., neural networks need to be enabled to have a memory.

This network type is quite different from other types of neural networks, thus I will briefly describe their intuition and implementation. Generally, RNNs can be viewed in two ways: 1) a neural network with an input, an output and a self loop, or 2) a sequential copy of the same network where the output of one is the input to the next network (Figure 3). The second version is often called the "unrolled" equivalent of the first version, where the sequential copies replace the self loop [67].

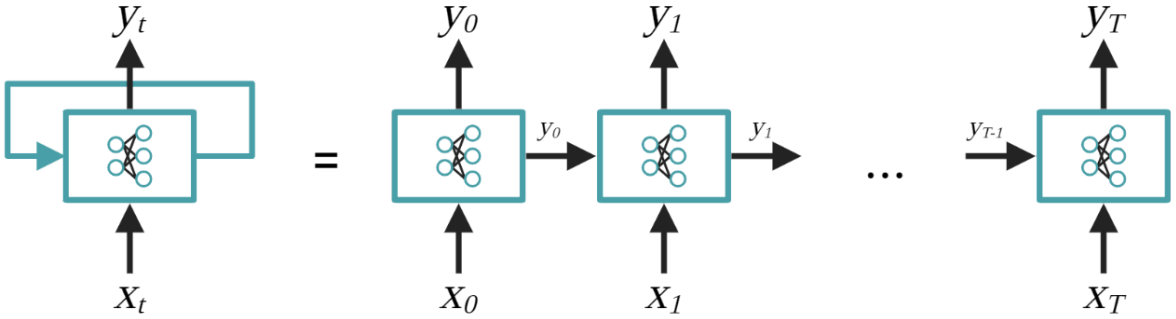


Figure 3: Unrolled representation of an RNN. An RNN has at any step t an input, an output and a self-loop that allows the memorisation of information from previous steps. The information flow through the network across all steps T can be visualised by "unrolling" the network. Here, the self-loops are replaced by sequential copies of the same network.

This loop-based information passing is also what makes RNNs suitable to handle inputs of varying length: The RNN is applied to one token of information (e.g. a word or a datapoint from a series) at a time [68]. What differentiates different types of RNNs from each other is the exact architecture of the network. While early RNNs struggled with memorising information for longer sequences, Long Short-Term Memory cells (LSTMs) [69] much improved this ability. Their success is based on a concept referred to as the *cell state* c_t as well as a series of so-called *gates*. The cell state represents the memory and it is passed on between loops in addition to the output, or *hidden state* h_t. The gates provide the option of manipulating the memory. They allow the network to filter what information is important enough to be passed on, what information, i.e. which "experiences", should be added in the current step and what to forget. In the following, $\odot$ denotes the element-wise multiplication. Essentially, the cell state at step t is computed as

$$c_t = f_t \odot c_{t-t} + i_t \odot g_t.$$

Here, f_t is the output of the *forget gate*. The forget gate receives as input the output h_{t-1} of the previous step as well as the data input x_t at the current step. This could be for example the next word in a sentence or the t-th measurement in a series of measurements. The forget gate applies a sigmoid function to these combined inputs that provides a number from $[0, 1]$ for each entry in the cell state, indicating how important it is to keep (values closer to 1) or forget (values closer to 0) the information. f_t is then applied to the previous cell state c_{t-1} by element-wise multiplication ($\odot$). i_t is the output of the *input gate*, which is again a sigmoid function indicating the importance of information carried in the input (h_{t-1} and x_t) and which

thus needs to be added to the cell state to be memorised for future steps. g_t is a vector of values to be added to the cell state which is first weighted by i_t using element wise multiplication. Unlike the other gates, the gate that produces g_t is not a sigmoid function but instead a hyperbolic tangent.

But not only the cell state is passed on to the next iteration, also the hidden state is transmitted, which is essentially a selective part of the cell state: Again, a sigmoid function is applied to the combined inputs (h_{t-1} and x_t) to evaluate which parts of the cell state to keep before passing them on. Then the cell state passes another hyperbolic tangent layer to limit its values to $[-1, 1]$. Then the re-scaled cell state is filtered using the output of the sigmoid.

Many variants of the LSTM have been released [70–72], one of them being the Gated Recurrent Unit (GRU) [73]. The GRU does not have a cell state and instead only relies on the hidden state to store and transmit information. It also uses fewer gates: An *update gate* and a *reset gate*. The update gate is a sigmoid function z_t that dictates how much of the current information is to be changed using the input x_t and the previous state h_{t-1}. The reset gate (also a sigmoid) decides how much of the information accumulated over past steps to forget, using as well the input and the previous state. The final update to the current state h_t is done with $h_{t-1} = (1 - z_t)h_{t-1} + z_t h'_t$, where h'_t is the proposed activation computed by a hyperbolic tangent using x_t, h_{t-1} and the output of the reset gate r_t. The GRU, unlike the LSTM, does not have a gate that controls to what extent the current information is exposed to the next step, but instead exposes it fully [74]. Additionally, the GRU does not use the forget and update gate independently to update the current state, which the LSTM does [74]. As a consequence of the lower number of gates, the GRU has less learnable parameters than the LSTM unit. They have been shown to achieve similar performances in sequence modelling when adjusted for similar parameter size [74].

Attention

RNNs were proposed for the task of sequence-to-sequence translation in 2014, in the same paper that introduced the concept of the GRU [73]. While these RNN-based Encoder-Decoder models already achieved good performance in sequence-to-sequence translation for longer sequences [65], they still require to encode all important information in one fixed length encoding from which alone the decoder had to construct the output.

Especially with long-term dependencies in the sequences, the existing models struggled to condense all the information required for a faithful translation [75,76]. To circumvent having to find one representation that encodes the entirety of the content as well as the dependencies between different elements, attention was introduced [76]. In the original proposal [76] (Figure 4), the key change was to introduce an *alignment model* a that was trained jointly with the encoder-decoder structure:

$$e_{i,j} = a(h'_{i-1}, h_j),$$

where h'_{i-1} is the previous hidden state of the decoder-RNN and h_j is the hidden state of the encoder-RNN at step j. The alignment model returns a score $e_{i,j}$ of how well the output of the decoder at step i matches the inputs to the encoder at position j. Thus in essence, it creates a score for all regions of the input sequence that indicates how important they are for the next prediction step of the decoder. These scores are then turned into weights with

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{T_x} exp(e_{i,k})}.$$

Here, T_x denotes the length of the input sequence. After this operation, there exist a weight for every input step j that indicates how important it is to predict the output at step i. But what are these weights applied to exactly? They weigh the hidden states of the encoder-RNN, which in conclusion means that this initial implementation of attention allows the decoder to access the hidden states of the encoder. The decoder uses the hidden states and their weights to calculate an encoding c_i. Unlike the encoder-decoder RNN without attention, it does not rely on a single encoding c of the input sequence for creating the output. Instead, at every decoding step, a step-specific encoding c_i is used. c_i is the weighted sum of the encoder's hidden states

$$c_i = \sum_{j=1}^{T_x} \alpha_{i,j} h_j.$$

There is one last key change to the original encoder-decoder-RNN that the authors made when introducing attention [76]: The encoder actually comprises two RNNs, one reading the input sequence in order, i.e. x_1, …, x_T, while the other reads it from back to front, i.e. x_T, …, x_1. The intuition behind this additional step is, that the hidden state $h_{t,forward}$ of the forward encoder contains information of all previous inputs x_1, …, x_t but not of the future

ones. However, the hidden stated $h_{t,backward}$ of the backward encoder encodes information on all futures inputs x_t, …, x_T. The combination h_t of $h_{t,forward}$ and $h_{t,backward}$ then features information on the surrounding area in both directions. This gives the alignment model a more information to score the area's importance on.

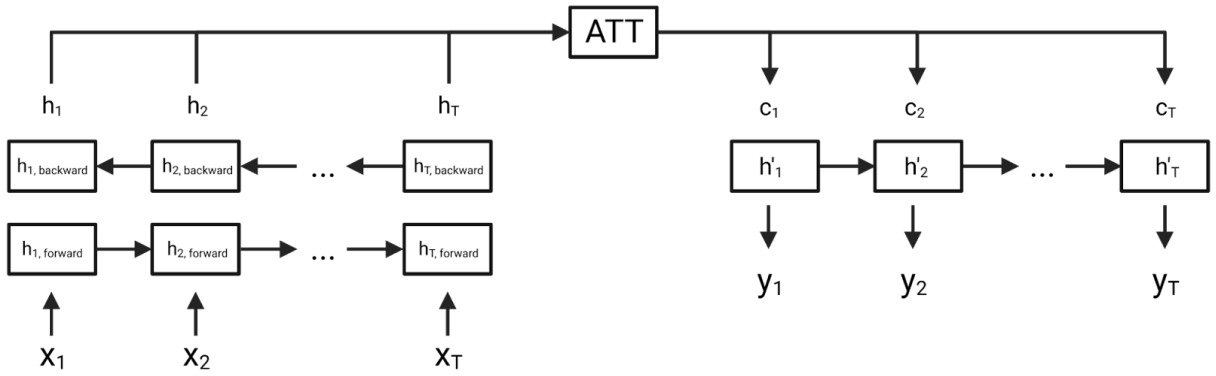


Figure 4: Attention. Illustration of the attention mechanism as proposed by Bahdanau *et al.* [76],c_t: cell state at step $t \in 1, \ldots, T$; h_t, $h_{t,forward}$, $h_{t,backward}$: combined hidden state, hidden state of the forward encoder and hidden state of the backward encoder, respectively, at step $t \in 1, \ldots, T$; x_t: input at step $t \in 1, \ldots, T$; y_t: output at step $t \in 1, \ldots, T$.

The disadvantage of this method is, that the decoder RNN does not only require access to a latent representation of the input but to all encoder hidden states as well. In sequence-to-sequence models this is mostly not a problem since the encoder and the decoder are not intended to be separated as they are typically used for end-to-end translation tasks. However, it is problematic if the decoder is to be used in a decoupled manner from the encoder or if the model is used for generative purposes. In the latter case, when sampling the latents, the encoder hidden states cannot be accessed since they are not known. Work has been done to decouple the decoder from the encoder in this regard, either by making the attention sampleable as *variational attention* [77] or by applying attention only once [78]. The latter essentially modifies the original attention such that there is not a separate encoding c_i for each decoding step but instead only a single encoding c. However that single encoding is a weighted sum of all hidden states, rather than the last hidden state h_t of the encoder. This approach lies somewhere in between no attention and the original attention of Bahdanau *et al.* [76] and is used in this work to decouple the encoder and the decoder.

1.1.2 Variational Autoencoders

All three projects presented in this thesis make use of Variational autoencoders (VAEs). Variational autoencoders (VAEs) were introduced by Kingma and Welling in 2013 [79] and are likelihood-based models, i.e. models which approximate the training data's probability density function [46,80,81]. Variational autoencoders comprise two deep neural networks: An encoder and a decoder (Figure 5). The encoder is a network that receives a data sample x as an input and maps it onto a latent space z. The decoder subsequently uses the latent representation z as input and attempts to produce a sample x' from the original data distribution. In case of a basic (non-variational) autoencoder, x' is trained to be an approximation of the original input x, using a reconstruction loss. Hence, the autoencoder learns a latent representation of the data that is typically lower-dimensional but contains the most important information about the input in a condensed way, making autoencoders especially interesting for dimensionality reduction tasks [82,83]. Given the non-linear nature of deep neural networks, this reduced representation is itself also not linear, differentiating it from other dimensionality reduction methods, such as Principal Component Analysis (PCA) [84–86]. Since the process does not require the samples to be labelled, it falls into the category of unsupervised learning tasks. However, the latent space of non-variational autoencoders is not necessarily continuous, meaning that close data points may or may not lie within close proximity in the latent space. While this is not a problem if the aim is to find a (reduced) numeric representation of the data, it does not allow data generation. For data generation, the latent space must be sampleable in a meaningful way, i.e. it must be continuous. This is achieved by modifying the models into variational autoencoders. Here, the encoding of an original data point x is not a single point z in the latent space, but instead each feature, i.e. dimension, of the latent space is modelled as a probability distribution [46]. A point is then sampled from that distribution and passed on to the decoder. To make these distributions easily describable, they are modelled as Gaussians [46]. Hence, the encoder needs to return only the means μ and variances σ of the Gaussians [46]. The continuous modelling of the latent space has to be enforced, the autoencoder will not realise this on its own, otherwise basic autoencoders would also provide a continuous latent space. To enforce this, the loss function used during training constitutes two parts: The reconstruction loss - just as in non-variational autoencoders - to ensure correct reconstruction of a given data point, and the regularisation term in form of the Kulback-Leibler (KL) divergence [46,87]. The latter ensures that the distributions returned by the encoder closely resemble a standard

normal distribution. Since random sampling makes backpropagation - and thus, training - impossible, the reparameterization trick is used to make the network trainable [79]. Up to this point, sampling from the learned latent space would approximate the data distribution of the entire training set. However, many situations demand a more guided generative process. Say one does not only want to generate a realistic looking image of a hand-written digit, but more specifically an image of the digit 2. To address such application cases, *conditional* VAEs were introduced [88]. Here, the label y of each training data point is provided to the encoder as well as the decoder during training. Subsequently, conditional generation is possible by sampling from the latent space *and* providing the label to the decoder. Since this requires a labelled dataset for training, conditional VAEs do not belong to the group of unsupervised learning mechanisms anymore.

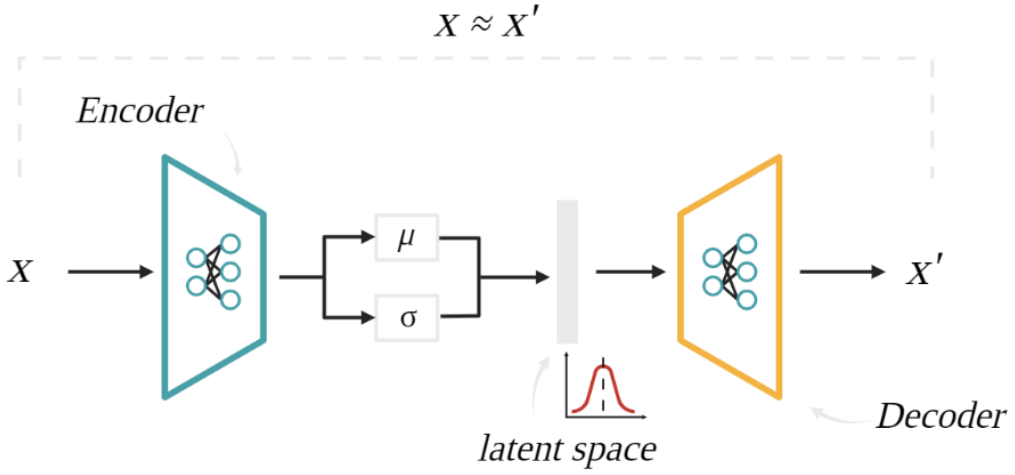


Figure 5: Variational Autoencoder. A sample x is encoded into a latent space by a network referred to as the encoder. The second network present in the VAE, the decoder, then attempts to generate a reconstruction x' from the latent representation. The model is trained such that the reconstruction x' closely resembles the original sample x. Since the latent space is modelled to be a multivariate Gaussian, the encoder is only required to output the means μ and variances σ.

1.1.3 Generative Adversarial Networks

Another type of generative model will be addressed in the first as well as the third project: Generative Adversarial Networks (GANs). In 2014, Ian Goodfellow *et al.* [47] introduced this type of generative model. GANs are implicit generative models, i.e. models that represent a sampling process of the data distribution [81,89]. Like VAEs, GANs consist of 2 neural networks: one responsible for the generation process and another auxiliary network (Figure 6). However, the way in which GANs work is substantially different from VAEs. The two neural networks that comprise the GAN architecture are referred to as the generator G

and the discriminator D [90]. The general goal is, that given a complex target distribution p_t - e.g., that of images of birds - the generator is a neural network approximating a transformation function that takes as input a sample from a simple distribution p_z (typically a Gaussian) and transforms it into a sample of the complex target distribution. In other words: It takes as input noise and generates as output for example an image of a bird. The target domain is of course not restricted to images, as will be demonstrated later. However, since the target distribution is unknown to G (and typically also to us), feedback on the performance is required that is not based on knowing the explicit formulation of p_t. This feedback is provided by the discriminator. The discriminator is a classifier that receives real samples $X = \{x_1, \dots, x_n\}$ from the real data distribution p_t and synthetic samples $X' = \{x'_1, \dots, x'_n\}$ from the generator's distribution p_g. It is then trained to discriminate between real and synthetic samples. If G perfectly reconstructed the target distribution, i.e., $p_g = p_t$, then D has no means to differentiate the two and performs best by guessing, achieving an accuracy of $\frac{1}{2}$. To approximate this ideal state, G and D are trained in an adversarial manner, playing a minimax game with the following loss function [90] :

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim p_t}(x)[log\, D(x)] + \mathbb{E}_{z\sim p_z}(z)[log\,(1 - D(G(z)))]$$

Here, $D(x)$ is the probability returned by the discriminator, that a sample x is real. Thus, the first summand $\mathbb{E}_{x\sim p_t}(x)[log\, D(x)]$ gives the expected log-probability that the discriminator recognizes a real sample x from the target distribution as such. For perfect classification of real samples, this becomes 0, for increasingly poor classification this approaches -∞. The second summand $\mathbb{E}_{z\sim p_z}(z)[log\,(1 - D(G(z)))]$ addresses the discriminator's ability to detect fake samples generated by G. For perfect classification of fake samples, i.e., $D(G(z)) = 0$, this term becomes 0, for increasingly poor classification it also approaches -∞. Hence, the goal of the discriminator is to maximise the sum, while the generator's goal is to minimise it.

After iterative rounds of training, new data instances can be generated by sampling from the noise prior and passing the samples through the trained generator network. For conditional sample generation, sample labels y can be provided to the generator as well as the discriminator during the training process using an additional embedding layer. The objective function then changes to a conditional form as follows [91]:

$$min_G max_D V(D,G) = \mathbb{E}_{x \sim p_t}(x)[log\, D(x|y)] + \mathbb{E}_{z \sim p_z}(z)[log\,(1 - D(G(z|y)))].$$

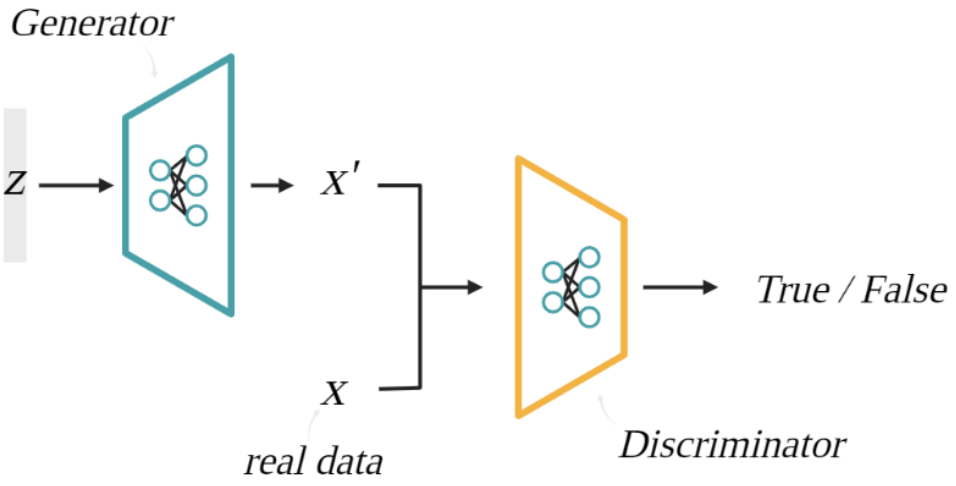


Figure 6: Generative Adversarial Networks. A generator network receives random noise z as input and is trained to produce synthetic samples x' of a target domain X the distribution of which is indistinguishable from that of the true samples. Another network, the discriminator, is trained to correctly differentiate between the synthetic samples and the real ones. The two networks are trained adversarially until the generator performs well enough to deceive the discriminator to an extent that it resorts to guessing whether or not a sample is real.

1.1.4 Diffusion Models

The last generative model I will introduce in detail is the diffusion model, which will be the central point of the third project. The idea of diffusion models was originally introduced as *diffusion probabilistic models* in 2015 by Sohl-Dickstein *et al.* [25], however, the method really became popular after a 2019 paper by Song and Ermon [26] introducing a similar approach called *noise conditional score networks* and a 2020 paper by Ho *et al.* [27], building upon the former two to introduce *denoising diffusion probabilistic models* (DDPM). Since these models are fairly new, there is still a lack of standardisation with respect to formalisation and implementation. To illustrate the connections better, I will give a brief overview on how the models introduced by the three papers are connected.

> *"The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data [...]."* — Sohl-Dickstein *et al.* [25]

Diffusion probabilistic models [25] and denoising diffusion probabilistic models [27] use a discrete noising process, that gradually adds Gaussian noise to a sample x_0 over $t \in \{0, \dots, T\}$ timesteps (Figure 7). The noise added to generate the next state x_t is depending on the previous noised sample-state x_{t-1} as well as a time dependent noise schedule - or diffusion schedule - $\beta_t \in (0, 1)$. Ho *et al.* formulate the noising step as

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I).$$

This is a recursive step, requiring the the computation of all previous timesteps $x_{0:t-1}$ in order to compute x_t. Applying the *reparameterization trick*, however, allows to directly calculate x_t from x_0. This reparameterization trick was originally introduced in Kingma and Welling's VAE paper [79] to allow the application of stochastic gradient descent despite the random sampling of the latent variable z. The general idea is to "outsource" the randomness to an auxiliary variable ϵ, essentially performing the following reparameterization:

$$z^{(i,l)} \sim q_\Phi(z|x^{(i)}) = N(z; \mu^{(i)}, \sigma^{2(i)} I)$$

$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}, \text{ with } \epsilon^{(l)} \sim N(0, I),$$

where i denotes the index of a sample in a dataset and l denotes the index of a Monte Carlo sample, i.e. a random sample.

For the diffusion step this means, that when defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, it can be rephrased as

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, \sqrt{1-\bar{\alpha}_t} I). \text{ [27]}$$

A detailed explanation on the stepwise transformation is given in [92]. Now, the noised sample at timestep t can be directly computed from x_0, since the β-schedule is known.

The generative ability of diffusion probabilistic models now comes from training a model p_θ, θ being the model parameters, that learns to reverse the diffusion process described above and, thus, allows to generate a sample from $q(x_{t-1}|x_t)$. Stepwise application of this model would then generate a sample from the original data distribution $x_0 \sim q(x_0)$ starting from Gaussian noise x_T.

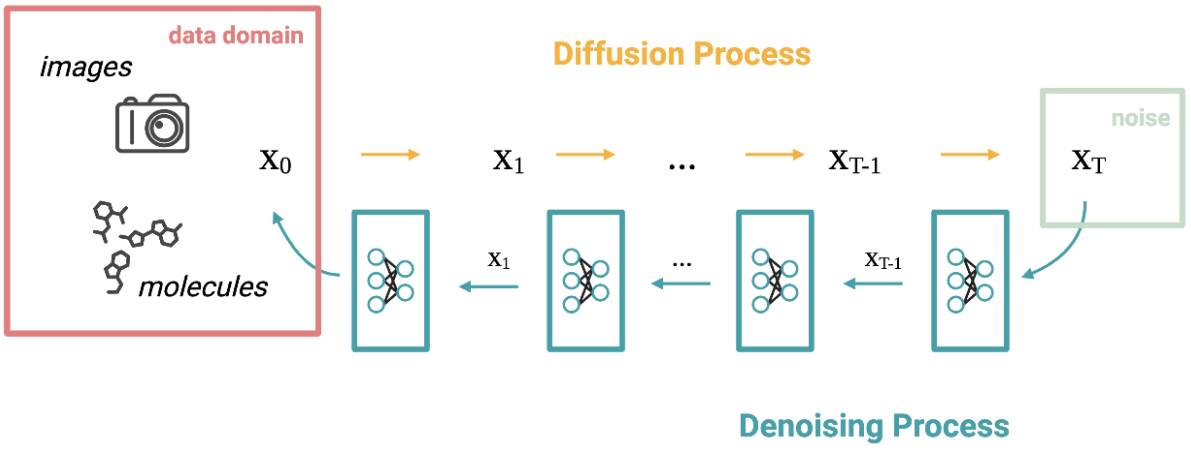


Figure 7: Diffusion Models. Data instances x_0 from a target domain (here exemplary images or molecules) are gradually noised over T time steps in a process called diffusion until at time point T the sample x_T approximates pure Gaussian noise. The model is then trained to perform stepwise denoising back from Gaussian noise to the original data domain.

Ho *et al.* describe this reverse diffusion process as a Markov chain

$$p_\theta(x_{0:T}) = p(x_T) * \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t).$$

Here, $p(x_T) = N(x_T; 0, I)$ and $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ with μ_θ and Σ_θ being the mean and the variance, respectively, of the conditional Gaussian parameterized by the model parameters and each conditioned on the level of noise at time point t. However, Ho *et al.* set $\Sigma_\theta(x_t, t) = \sigma_t^2 I$, with $\sigma_t^2 = \beta_t$ - a known constant from the variance schedule - which leaves only μ_θ to be learned by the model. As an objective function for the training of p_θ, the variational lower bound was proposed [27], which is the same objective used to train variational autoencoders. This is possible, since the noising process q and the denoising process p_θ together form a variational autoencoder [27]. This variational lower bound can be expressed as a sum of losses [25,93], with each summand being the loss at a single time step t

$$L = L_0 + L_1 + \ldots + L_T$$

with $L_0 = -\log p_\theta(x_0|x_1)$ and all other terms L_1, ..., L_T being the Kullback-Leibler divergence between two Gaussians, which is a distance measure for the similarity of two Gaussian distributions. Given that x_t can be calculated directly from x_0 due to the above described reparameterization trick, and the just stated property that the loss is a sum of losses

for each t, the loss term L_t can be optimised randomly during training requiring only x_t and x_0.

Simplifying implementation and improving sample quality, Ho *et al.* found further that the mean μ_θ, which the denoising model is trained to predict, can be further reparameterized in terms of a noise predictor ϵ_θ [27]. Thus, finally, the loss term L_t for $0 < t \leq T$ becomes $L_t = \mathbb{E}_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2]$ and the model now learns to predict the noise that was added at step t, rather than the sample itself. Note that $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$, after the reparameterization trick.

The diffusion models as described above are essentially describing the noising-denoising procedure in analogy to denoising autoencoders: a sample is noised (encoder, forward diffusion) and then reconstructed by a denoising process (decoder, reverse diffusion). Even the objective function is inspired by autoencoders.

As introduced in the beginning of this section, there are different ways to describe diffusion models. Another way that was developed independently is based on score matching [26,94]. Recent work indicates that both approaches are likely to be different descriptions of the same model type [27,95]. In the score-matching approach, a model $s_\theta(x)$ is trained to learn the score function $\nabla_x log\, q(x)$, i.e. the gradient of the log density of the data distribution $q(x)$ with respect to the parameter x [81] (figure 8).

Figure 8: Score function. Red vertical lines indicate data density with close lines representing low density and far apart lines representing high data density. The score function is indicated by the vector field, where longer vectors indicate a stronger gradient and shorter vectors indicate a smaller gradient. Sampling from the distribution using the score function and Langevin dynamics would randomly initialise a sample and then move it along the vector field for *k* steps.

After training $s_\theta(x)$ an additional method is needed to sample it and generate new data points. This method is called Langevin dynamics and allows to sample $q(x)$ when only knowing $\nabla_x log\, q(x) \approx s_\theta(x)$ [26]. Essentially, an initial sample x_0 is generated from a prior, which can be arbitrary. Then the sample is moved along the vector field represented by the score function to areas of higher log density of the data. This approach of data generation

comes with a complication, that is discussed by Song *et al.* [26]: During training, the score function is estimated by the model $s_\theta(x)$ by learning to predict the scores of real data points from the training set. Naturally, the model does not perform well for low-density areas of $q(x)$, given the small number of training examples. However, when picking a random initial sample x_0 to start the Langevin sampling, x_0 is likely to be taken from a low-density area, where the approximation of $\nabla_x log\, q(x)$ is poor. This is where the strong parallel to the diffusion approach described above comes in: The training of $s_\theta(x)$ is improved by perturbing the data using Gaussian noise of different variance. The motivation behind this is that adding noise will alter the samples such that they enrich the otherwise low-density regions. As illustrated in [26,81], adding a lot of noise facilitates populating those low-density areas. However, it also changes the original data quite drastically. Hence, much like in the first approach, different noise based on a variance schedule is added. To stay with the notation used above for better readability, the noise added is sampled from $N(0, \beta_i I)$ for $i = 1, \ldots, L$ and $\beta_1 < \beta_2 < \ldots < \beta_L$. For every i a new data distribution $q_{\beta_i}(x)$ is created by sampling x from $q(x)$ and adding $\beta_i z$ with $z \sim N(0, I)$ [81]. Then, a model conditioned on i can be trained to approximate the score for each perturbation:

$$s_\theta(x, i) \approx \nabla_x log\, q_{\beta_i}(x).$$

The sampling procedure is slightly different, now that a different model exists for each perturbation step. Song *et al.* [26] proposed *annealed Langevin dynamics*, adapting ideas from simulated annealing. Here, a random initial sample x_0 is generated from a prior and Langevin sampling is applied using $s_\theta(x, i)$ for $i = L, L-1, \ldots, 1$. The final sample from $q_{\beta_L}(x)$ is used as input to the next sampling step using $s_\theta(x, L-1)$ and generating a sample from $q_{\beta_{L-1}}(x)$. This is repeated until $i = 1$, producing the final sample which, assuming successful training, approximately follows the original data distribution.

While the sample quality of both approaches is high, the achieved negative log-likelihood was still subpar to that of other state-of-the-art generative models. Additionally, the sampling time is long in comparison to, e.g., GANs, due to the iterative denoising process of T steps. The former issue was improved by using cosine noising schedules and introducing an approach to learn the variance instead of fixing it as mentioned above [93]. The latter issue was addressed with *denoising diffusion implicit models* (DDIMs) [96], which allow to sample

from the model with $S < T$ steps while training with T steps by using non-Markovian diffusion. DDIMs further exhibit sample consistency: starting from the same initial noise x_T, but applying the reverse diffusion process with different numbers of steps, produces very similar outputs. This indicates meaningful latent spaces and provides the option for interpolation [96].

As it is the case with most deep-learning methods, also diffusion models have been originally developed on image data. A more generally applicable approach, named *latent diffusion models*, was introduced by Romach *et al.* in 2022 [28]. Here, the original data is first embedded using an Autoencoder and the diffusion model is then trained on the latent representations of the data. Although the paper showcased this approach again on image data, this opens up the possibility to use diffusion models with any kind of data that can be embedded with an Autoencoder. In the third project presented in this thesis, I used these models to generate molecules using molecule embeddings from a VAE for the training of the latent diffusion model.

As mentioned before in the context of VAEs and GANs, a critical aspect of generative modelling is the conditional generation of samples. Sohl-Dickstein *et al.* already discuss the topic of conditioning the diffusion process in their 2015 paper [25] and Dhariwal and Nichol applied it with much success in their 2021 paper *Diffusion Models Beat GANs on Image Synthesis* [97]. The idea is to guide the diffusion process of a trained diffusion model towards a target distribution using a pre-trained classifier. To allow guidance throughout the denoising process, the classifier must be trained on noisy samples and conditioned on the timestep. During the diffusion steps, the samples are passed to the classifier and the gradients of the classifier's conditional log-probability are then used to manipulate the mean used to sample the next timestep. Dhariwal and Nichol could also show that the success in fidelity of the generated samples could be strengthened by tuning the signal of the classifier. To this end, they scaled the gradient with a constant. What is advantageous about this approach is that the diffusion model can be trained on an unlabelled dataset and only the classifier requires training on labelled data. However, the fact that the classifier must be trained on time-conditioned noisy samples adds an additional complexity layer to its training.

1.2 Sharing Transcriptomics Data with Consideration of Patient Privacy

Training an AI model requires data. To accumulate enough data for a successful training process it often needs to be shared, which prompts the necessity for privacy protecting mechanisms, if the data is sensitive. This is the case for many modalities of medical data, among those, transcriptomics data.

1.2.1 Transcriptomics Data

In a living cell, the information on how to build the cell's proteins is stored in the DNA [98]. These proteins are involved in all major actions that a cell can take, ranging from metabolism to the cellular structure and the cell's response to external stimuli [99]. To create the proteins, building instructions have to be copied from the DNA (Figure 9). These copies are made of RNA [98] and the process of generating them is referred to as transcription [100]. During transcription, the part of the DNA to be copied is unwound, opening up its native helical structure. The hydrogen bonds between opposing nucleotides of the two DNA strands are broken and an enzyme called *RNA-polymerase* creates a complementary strand of RNA nucleotides [101]. The entirety of such RNA molecules that occur in a cell is referred to as the transcriptome [102,103]. The transcriptome comprises different types of RNA that hold varying biological functions: messenger RNA (mRNA) carries the information on the amino acid sequence of proteins and it is read and translated into the proteins by ribosomes [104]. Other RNA-types do not encode proteins but are involved in regulating their production. Among these are transfer RNA [104], ribosomal RNA [104], microRNA [105], long non-coding RNAs and others [106].

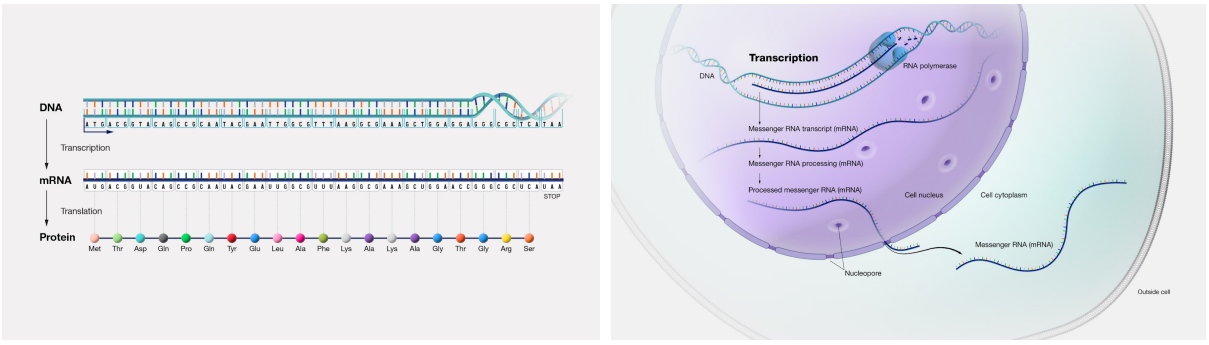


Figure 9: Transcription. Left: The DNA encodes the information needed to produce proteins. The information is copied from the DNA into a strand of RNA by Transcription. Proteins are then built using the information contained in the RNA during a process referred to as translation. Right: A more detailed view of the transcription process. RNA-polymerase

generates a messenger-RNA transcript from a DNA-segment and the RNA is transported to the cytoplasm for downstream translation. Courtesy: National Human Genome Research Institute [107].

Upon stimulation, i.e. due to environmental factors or disease, cells have to adapt the composition of their proteins, producing more of some and less of others to react to the stimulus [108]. This response is reflected in the amount of mRNA copies that are created for the different protein-coding genes, or in other words, it changes how strongly a gene is expressed. Hence, quantifying gene expression is an important approach in biomedical research in order to study and understand how a cell, a tissue or an entire organism responds to different challenges [109,110]. Over the years, different technologies have been developed to quantify the transcriptome [111–113]. The data used in this work was produced using a technique called sequencing by synthesis on a sequencing platform by Illumina [109,114]. Prior to the actual sequencing, the RNA must be isolated from the biological sample. Then, the RNA must be cleaned to remove RNA-types that are not to be sequenced, e.g. ribosomal RNA. Subsequently, a complementary DNA (cDNA) library is generated from the RNA (Figure 10). This process relies on an enzyme called *reverse transcriptase* that uses an RNA molecule as a template and constructs a complementary piece of DNA [115].

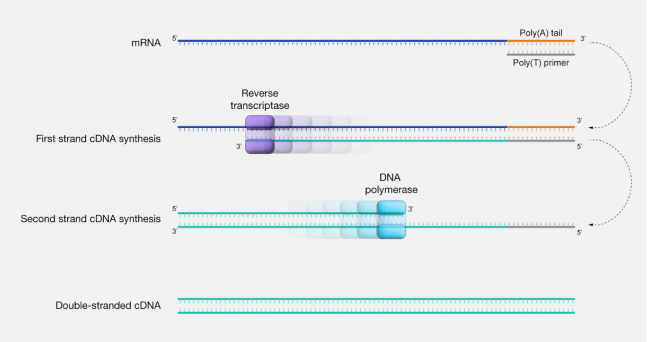


Figure 10: Copy-DNA. A complementary DNA-strand is created from an RNA-template by the enzyme reverse transcriptase. For double stranded copies, DNA-polymerase generates the second DNA-strand, complementary to the product of the reverse transcriptase. Courtesy: National Human Genome Research Institute [107].

This is the reverse process of the transcription described above. The cDNA is then amplified to enrich the amount of material for sequencing. During this step, various attachments are made to the cDNA pieces, among those a short sequence that allows the cDNA to bind to the glass slide on which the sequencing is performed. Another added piece is the index, which tags all cDNA molecules that belong to the same sample. After the single-stranded cDNA has bound to the glass slide (Figure 11), a process referred to as bridge amplification generates spatial clusters of both directions (forward and reverse strands) of the cDNA molecule. After

the clusters are generated, the reverse strands are cut off, such that each cluster now contains many copies of the same molecule. Then, a complementary strand is created for each cDNA. Special types of nucleotides are used during this step, where each base is marked with a different fluorescent colour. One nucleotide is added per iteration and at the end of each iteration, the molecules are excited with light to measure the fluorescent signals. These signals are recorded and allow the identification of the base added in each step for all cDNAs. While sequencing by synthesis is a very accurate sequencing technology, chances of errors increase with longer sequences. Thus, the gene copies are not sequenced in their entirety but cut into smaller sequence pieces of not more than a few hundred nucleotides. After identifying the RNA sequences of these *reads*, they are then mapped back to a reference genome and for each gene, the number of copies aligned to it can be counted. This eventually yields quantitative information on the expression strength of any gene, based on its sequenced copies.

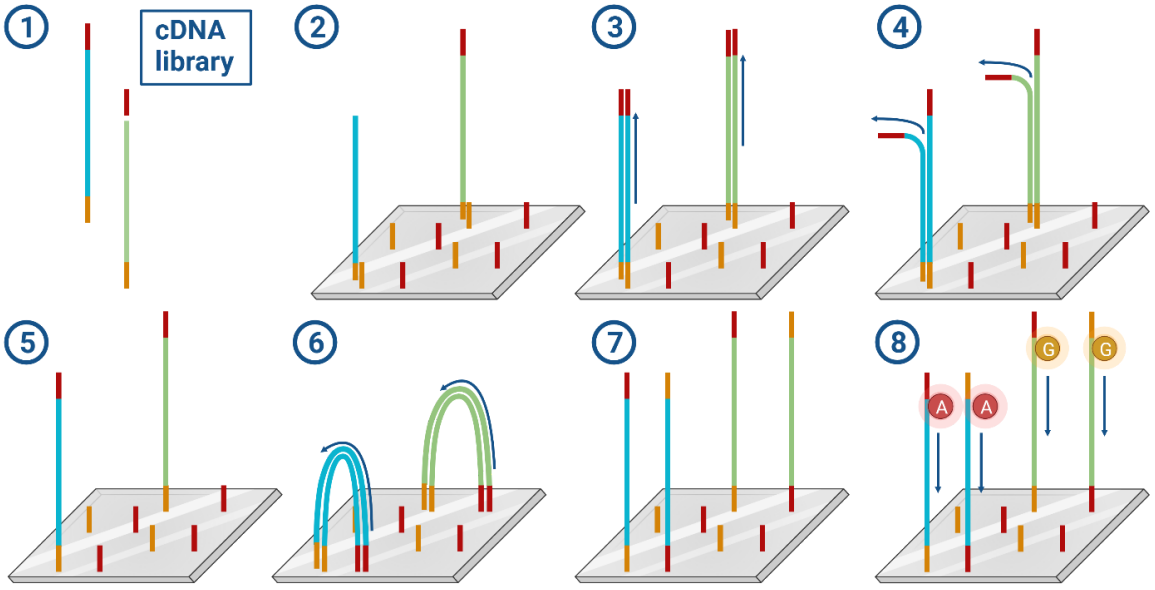


Figure 11: Sequencing by synthesis. The cDNA library (1) binds to the glass slide (2) and a complementary strand is synthesised (3). The original cDNA-strand is removed (4) leaving only the newly synthesised strands, which are fixed to the glass slide (5). Local clusters of the same molecule are generated via bridge amplification (6 & 7). The sequences of the clusters are measured by synthesising the complementary strands using fluorescently labelled nucleotides (8).

RNA-sequencing can happen at different resolutions (Figure 12). In bulk RNA-sequencing [116], RNA is extracted from a collection of cells, e.g. a tissue sample. The copy numbers yielded by bulk RNA-sequencing are thus average values across the cells that were present in the sample. Another, more recent, approach is single cell RNA-sequencing [116,117]. Here,

individual cells are separated and their RNA is isolated prior to sequencing. The copy numbers are then specific to a given cell, increasing the resolution of the data. While single cell RNA-sequencing allows a much more fine-grained and differentiated analysis of the transcriptome, it is also more costly and generates extremely high-dimensional data that is not easy to handle. The choice of whether to use single-cell or bulk sequencing depends on the research question.

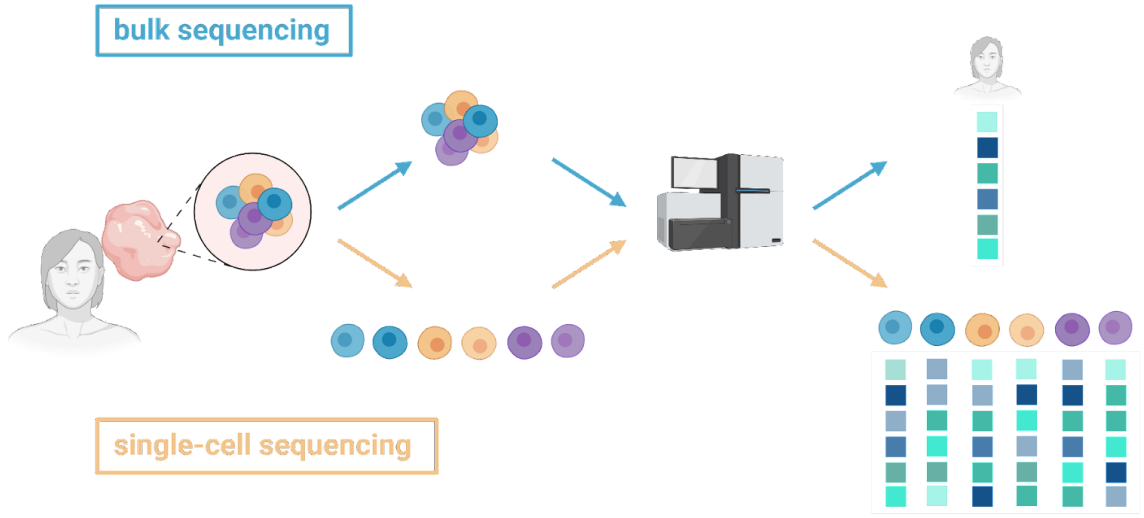


Figure 12: Bulk-sequencing versus single-cell-sequencing. During bulk-sequencing (top path), cells from a sample are sequenced without separating them first, yielding average quantities for each gene across the cells contained in the sample. In single-cell-sequencing (bottom path), the sample's cells are separated first and the transcriptome of each cell is measured individually, yielding gene-expression profiles for each of the cells present in the sample.

Sequencing the transcriptome under different stimuli is a common approach to investigate the function of genes. Functional genomics aims to identify the different roles of a gene product under varying conditions, pathways that a gene is involved in and groups of genes that form a functional entity and are often co-regulated together. Transcriptomics has been applied to answer a plethora of research questions. In Oncology, RNA-sequencing data has been used to classify different cancer types [118–121], to assess drug susceptibility [122,123] and to identify biomarkers [124]. RNA-sequencing and subsequent analyses of the transcriptome also allow the detection of different splice isoforms [125] as well as the identification of expression quantitative trait loci (eQTLs) [126], which are variants in the genome that impact the expression levels of one or more genes [127]. It has also been used for the diagnosis of rare diseases [128]. However, in order to maximise the knowledge extracted from transcriptomics data and utilise the insights to advance the understanding of diseases, the data often needs to be shared, which poses risks to the privacy of the patients.

1.2.2 Privacy Issues accompanying the sharing of transcriptomics data

As we discuss in our review paper [15], functional genomics analyses often rely on transcriptomic data. To conduct such analyses, the data many times requires sharing. This can be for various reasons, such as in the case of multi-party studies where the data was acquired at different sites and is intended to be centralised for analysis. Other reasons are sharing the data for the reproducibility of results, reusing a data set to answer a scientific question other than the one it was originally sampled for, accumulating data for machine learning purposes - especially for minority groups to balance the data classes and avoid biases [129] - or to advance personalised medicine [130,131], an endeavour that requires access to various medical data modalities in order to make an integrative decision. Hence, the reasons to share the data are vast, however, the implementation is often cumbersome: Patient privacy must be maintained but makes free sharing of the data precarious and additional steps have to be taken. The main threat to patient privacy in this context is subject re-identification [132,133], i.e. identifying the individual from which the data was sampled. 30-80 independent Single-Nucleotide-Polymorphisms (SNPs) are sufficient to uniquely identify an individual [133], allowing for the construction of SNP-fingerprints [134]. Such fingerprints can be extracted from publicly available genomic sequence data, often associated with meta data on medical conditions. Given an individual's DNA, their barcode can be matched to those extracted from public data. This threat not only resides on the level of sequence data, but consequently translates to all data modalities that allow the inference of SNPs. This is also the case for transcriptomics data. Genetic variants in expression quantitative trait loci (eQTLs) impact the expression levels of genes and are therefore reflected in the transcriptomic data. In a 2012 paper [18], the authors showed that SNP-fingerprinting was possible using the effects of eQTLs on transcription. Thus, measures must be taken to protect patients and study participants from re-identification. To this end, different methods exist with varying levels of security [15], reaching from low security when sharing the data freely, over medium security when restricting the access to the data, but allowing its download, high security for either restricted access without the option to download the data or making the data Safe-Harbor-compliant before sharing, to very high security when applying data sanitisation, fully homomorphic encryption, secure multi-party computation or differential privacy (Figure 13).

While these high and very-high security options exist, their application is not ideal. Controlled data access without the option for download restricts the analysis framework to

that offered by the data provider. From a genomics perspective, compliance to Safe-Harbor not only often removes meta information important for the analysis, but it is also not fully protecting the patients from re-identification. This is mainly because the Safe-Harbor regulations only apply to meta data and not the transcriptomic data itself. Thus, any re-identification threats posed by the transcriptomic data remain [135]. Safe-Harbor [136] and its successor the EU-US Privacy Shield [137] (both principles to uphold EU data privacy standards in the United States) have by now been overturned by the European Court of Justice [138,139]. However, Safe-Harbor compliance is still listed as a personal health information privacy standard by the Health Insurance Portability and Accountability Act (HIPAA) [140].

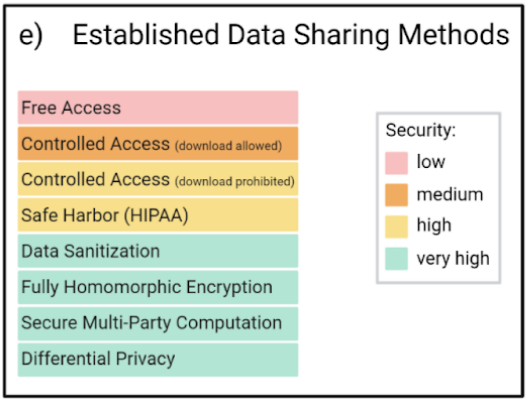


Figure 13: Security of data sharing methods. Various data sharing methods exist and they come with different levels of security for the data. Granting free access to raw data allows unlimited extraction of information held in the data and therefore provides low security (red). Increasing security can be achieved by controlling the access to the data (orange), restricting the ability to download it or removing particularly sensitive information types (yellow) Very high security guarantees can be made with extensive data sanitisation, encryption or differential privacy (green). This figure is taken from the figure panel 1e from [15] Copyright (c) 2021 Marie Oestreich, Dingfan Chen, Joachim L. Schultze, Mario Fritz, Matthias Becker published as Creative Commons — Attribution 4.0 International — CC BY 4.0.

Data sanitisation for transcriptomic data often resides on the raw read level as for example the method proposed by Gürsoy *et al.* [141], which replaces genomic variants of the individual with that of the reference genome. This allows sharing of the sanitised data, however it does not provide a statistical privacy guarantee and does not operate on the read count level.

More recently, there has been research into alternative approaches such as sharing-free solutions and private synthetic cohorts. Sharing-free solutions, such as federated learning [142] or swarm learning [143], entirely relinquish the idea of sharing the data and instead share machine-learning models that have been trained on them. Swarm learning takes this

concept another step further by entirely eliminating any centralised instance in the learning process that could pose an attractive target for an attack [143]. While this is only an option for the machine learning context, private synthetic cohorts offer a complimentary solution [144–146]. Here, the goal is to generate fake data that follows the same distribution as the original and yields the same results when analysed, but does not reveal identifying information on the subjects of the original data cohort. We pursued this idea in the ProGeneGen project described further down, using deep generative modelling.

1.3 Machine Learning in Drug Development

The drug development process and its accompanying challenges have been a topic of discussion for a long time. Already in a report from 2004 titled “Challenge and Opportunity on the Critical Path to New Medical Products” [147] the United States Food and Drug Administration (FDA) discussed the inefficiency of the drug development process and the urgent need for new methods that allow increased speed and better pre-selection of candidates, to stop the trend of increasing costs and yet decreasing success in the identification of new medical products. They identify the driving issue to be the growing gap between the proposal of compounds and their application, which is mostly hindered by an insufficient prior assessment of the compounds’ effectiveness and safety. This leads to immense costs in the earlier stages of the development process which are then not rewarded by a commercial product, making development of new drugs less and less sustainable. One of the directions proposed in the report to tackle the problem is the use of computational methods. Since then, many computational techniques have been developed in the pharmaceutical context, numerous ones based on ML [148–153]. Machine learning in general has found increasing application in almost all areas of life [154] and that includes pharmaceutical research and the pharma industry. Various pharmaceutical companies and consultant agencies are advertising the use of A.I. for drug development on their websites [155–159]. The capability of machine learning models to traverse extremely large data spaces and to extract their patterns makes them so interesting for the tasks posed in drug development, since the data that has been collected is becoming too immense to be analysed by humans in its full complexity [153,160,161]. This poses an opportunity for synergetic work between ML and researchers: The models allow the fast and exhaustive handling of the data as well as general pattern extraction, while the experience of specialised researchers adds

expertise for subsequent finetuning and evaluation as well as knowledge integration on outlier cases.

1.3.1 Molecule Autoencoders and Their Optimisation Towards Quality and Resource-Consumption

To train any deep generative model on the task of molecule generation, suitable molecular representations are needed. Many computer-readable formats exist, but they are often not suitable for training. But what characterises a good representation?

The propagation of input through a neural network is based on mathematical operations, hence a numeric representation is required. For data modalities that are not numeric, e.g. text, this can be enforced using one-hot encoding: Here every token - that can be a symbol or a combination of symbols with a fixed meaning - is represented by an index. The one-hot encoding of a data instance is then a 2-dimensional matrix, one dimension representing the alphabet (set of unique tokens) and the other dimension representing the number of tokens present in the data instance. As an example, to one-hot encode english words, the set of tokens, i.e. alphabet, are the 26 letters a-z (assuming we are writing only in lowercase). To encode the word “hello”, a vector of length 26 is constructed for each letter, where every value is zero except for the index that represents the current letter in the alphabet, which is set to one (figure 14). While one-hot-encoding allows the numeric representation of non-numeric data, it inflates the representation by a factor equal to the size of the alphabet.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
h	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0

Figure 14: One-hot-encoding. Exemplified one-hot encoding of the word “hello” using an English, all lower-case alphabet. For each letter of the word, the index of that letter in the alphabet holds the value 1 whereas all other indices hold the value 0.

Another desirable characteristic of a molecular representation is continuity, i.e., molecules that are similar in chemical space should have similar representations and the transitions between them should be reflected in their encodings. Thus, encodings that are numerically close to each other also represent chemically similar molecules, providing chemical meaning

to proximities in the encoding space. Further, if their representations are unique, that would ease correct decoding of the molecule by avoiding multi-mapping ambiguity. Additionally, having a fixed length would improve compatibility with various model types, ideally as short as possible while maintaining sufficient information to minimise resource requirements.

Most existing molecular representations do not follow these rules. For example, string representations that encode molecules using characters are non-numeric and differ in length. A common string representation is the simplified molecular-input line-entry system (SMILES) [162], which can be found in most databases. SMILES are generated by depth-first graph traversal over the molecular graph, encoding encountered atoms, bonds, branches and charges with character symbols. Given the multitude of starting points for the traversal, SMILES are not unique [163]. Additionally, a random arrangement of SMILES tokens does not necessarily translate to a valid molecule [163]. For example, this can happen when paired tokens, such as parentheses which are enclosing branches, are lacking their counterpart. The issues of uniqueness and validity are mitigated by using self-referencing embedded strings (SELFIES) [164]. SELFIES guarantee validity even when the tokens are randomly generated, however, they remain non-numeric and vary in length. A more detailed explanation of string-based molecule representations including their advantages and disadvantages can be found in the appendix. An example of a molecule's SMILES and SELFIES representation is illustrated in figure 15.

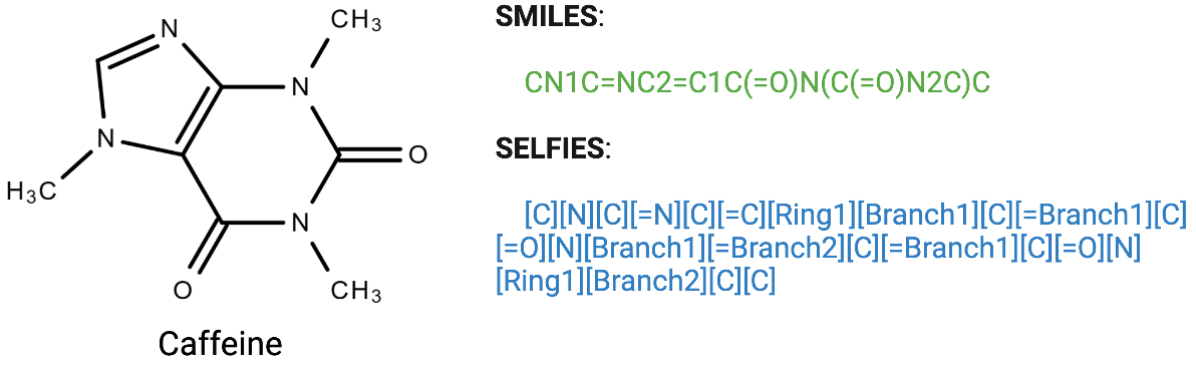


Figure 15: SMILES and SELFIES. SMILES and SELFIES strings exemplary illustrated for the molecule *caffeine*.

Another common molecule representation are fingerprints. These are fixed-length numeric vectors constructed based on the presence of substructures or atom neighbourhoods, e.g. Morgan/ECFP4 [165] fingerprints or MACCS keys [166]. While they are of fixed length and numeric, they suffer from multi-mapping issues and resulting non-invertibility: Several

molecules may map to the same fingerprint which makes correct reconstruction impossible [167].

When it comes to structure representation, the most intuitive choice are probably molecular graphs. Here, nodes and edges are utilised to depict atoms and bonds, respectively [168]. Atom- and bond-types can be distinguished by storing representative values in the nodes and edges. However, molecular graphs are neither numeric nor of fixed length and they cannot be easily one-hot encoded, given their different components (node-types, edge-types, adjacencies). Graph neural networks [169] were specifically developed to make graphs processable for machine learning.

The molecular structure can also be captured with 2- and 3-dimensional images. Especially voxel-based representations allow a very high level of detail: different atoms and bonds can be depicted using voxel values, bond lengths and angles can be used for correct spatial arrangement [170]. However, the high dimensionality inflates the size of each data instance, making the training more memory intensive than with other representations. Additionally, these representations are typically sparse, with most pixels/voxels representing the empty space around the molecule [170]. There have been approaches to utilise the available space by modelling the atoms and bonds as 2D- or 3D objects rather than points and lines, respectively [170]. While this reduces sparsity, the high data dimensionality remains.

To address the shortcomings of existing molecular representations, (variational) autoencoders have been used to learn an embedding of the molecules that can then be used to train downstream machine learning models [19–21,171]. These embeddings are numeric, of fixed length, unique and contain enough information to reconstruct the molecule. While these embeddings theoretically also allow for continuity, this aspect is often not sufficiently analysed and the impact of different autoencoder architectures on the embedding quality has frequently been ignored. A systematic analysis of how architecture impacts encoding quality is required in order to optimise the embedding for downstream tasks. However, optimisation should not only be geared towards embedding quality but also towards resource consumption: Training machine learning models consumes a significant amount of resources. In face of global warming and constantly reducing resources, the concept of green AI is increasingly endorsed, promoting to reduce the energy footprint of AI models [22,23].

1.3.2 Compound generation using deep learning

While machine learning can be involved at any step of the drug development process, such as the identification of drug targets or desired molecular properties, the proposal of hit compounds as well as their optimization to leads, or even study design, quality control and manufacturing pipelines [32,33] (Figure 16), two tasks are of particular interest in this thesis: 1) models that, given a molecule, predict a molecular property (e.g. solubility, toxicity, synthesisability) and 2) models that, given a property, generate a molecule that exhibits this property. The former are discriminative models such as regression models or classifiers, the latter are generative models. There are various attempts at *de novo* molecular design using deep generative modelling. They can generally be grouped into those that construct the molecules iteratively, be it atom by atom [172] or by adding substructures [173], and those that generate the molecules in full [171,174]. While iterative approaches indirectly mimic synthesis and therefore may address synthesisability by design, their success is highly dependent on the exact modelling procedure: Models that generate molecules atom by atom often produce invalid intermediates and end products by violating chemical validity on a super-atom level [173]. This can be avoided by either defining comprehensive rules [20,175,176] or using molecular substructures as building blocks instead [173]. The latter typically requires the construction of an extensive substructure alphabet.

In the context of generating compounds based on provided properties, a conditioning of the generative process is required. Most methods to incorporate conditions into the training process rely on training with paired data, where the possible conditions and their values for the different training instances are provided to the model during training [171,177–181]. This comes at the great disadvantage of having to retrain the model whenever conditions are added, removed or combined. This not only hinders flexibility but also does not align with the prospects of sustainable model design and green AI and is thus an issue that needs addressing.

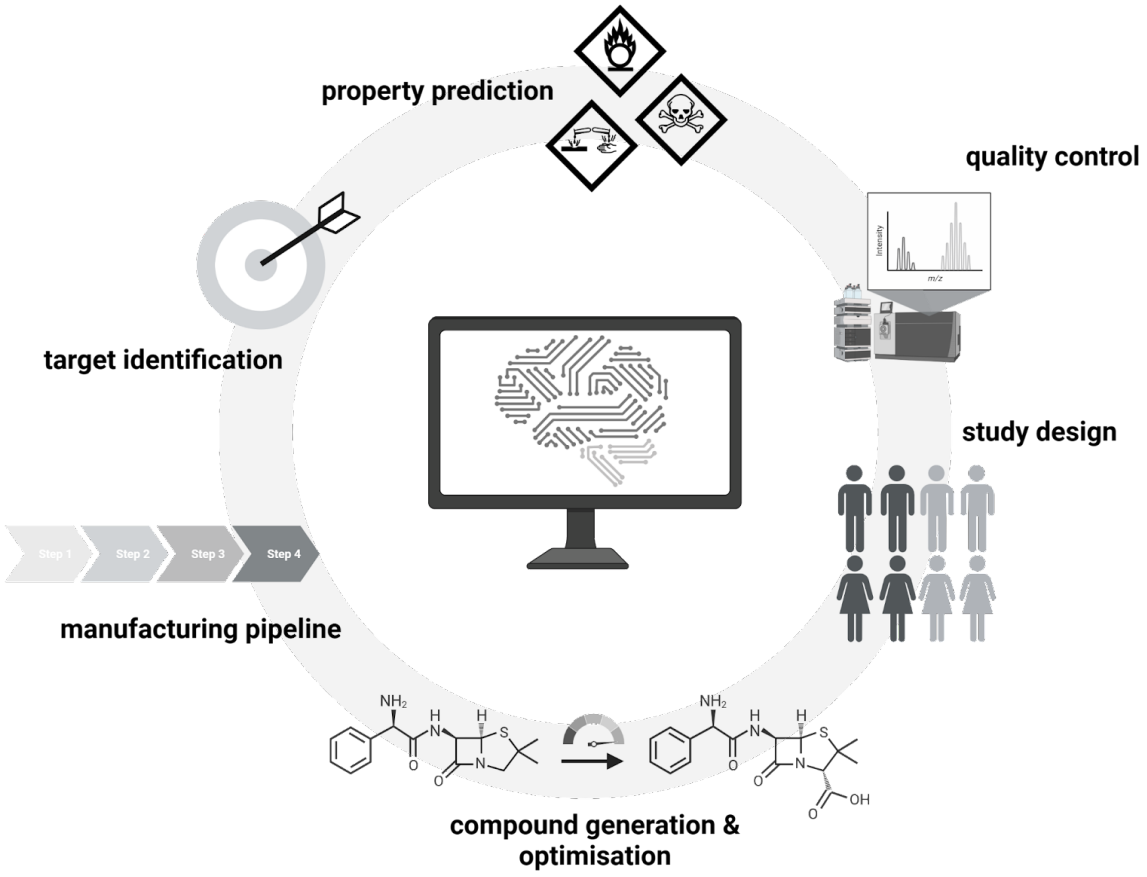
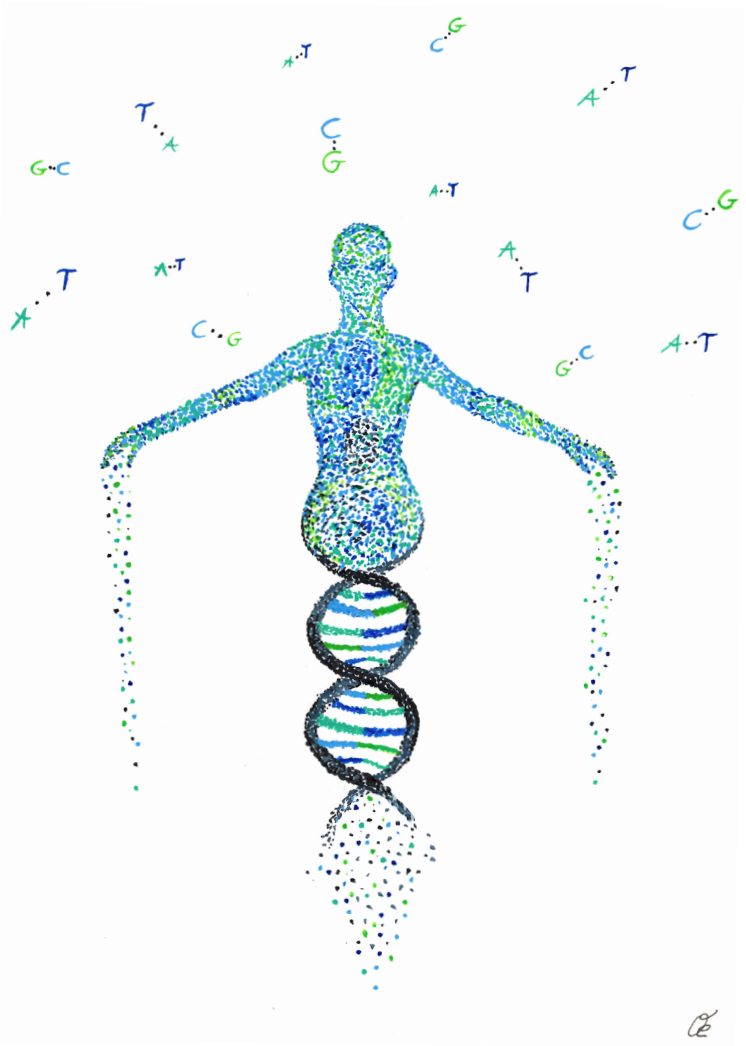


Figure 16: AI in drug development. The involvement of artificial intelligence in various aspects of the drug development process.

1.4 Goal of the Thesis

In summary, the main goal of this thesis will be to illustrate the contribution of generative modelling to the field of systems medicine: 1) I will specifically introduce its potential in the context of data privacy of high-dimensional medical data spaces, exemplified on transcriptomics data. 2) I will further delineate the capability of generative models on the task of property-driven compound generation. Here, the first goal will be to address the issue of molecular data pre-processing with auxiliary models, i.e. autoencoders, and their optimisation towards encoded information content as well as resource management in the context of green AI. The second goal is then to illustrate the use of diffusion models for the conditional generation of molecules and how to avoid the issue of retraining while maintaining flexibility in terms of adding or removing properties.

2. **ProGeneGen** - Protecting Genetic Data with Synthetic Cohorts from Deep Generative Models



2.1 Introduction

Genomic data is among the most personal data that can be collected from an individual. Hence, the information it contains is highly sensitive and must be handled with care as to not violate the privacy of the individual. At the same time, the information is extremely valuable to research, since it allows us to better understand disease mechanisms and develop new treatments. To facilitate such crucial research attempts, we must share the data with colleagues, collaborators, analysis experts and doctors. Various different methods have been developed to share data associated with the genome. However, there is an observable trend where privacy is traded for utility, making the very high security methods difficult to work with. Thus, research in this field is still ongoing and two approaches have peaked particular interest in recent years: 1) sharing-free solutions such as swarm learning [143] or federated learning [142,182], where not the data itself is shared but instead insights gathered from the data during model training, and 2) synthetic data approaches, where a "fake" dataset is generated that is private and thus freely shareable but that exhibits similar properties as the original data. These two approaches are not competitive but rather complementary since they solve different problem scenarios. Swarm and federated learning are the go-to solutions when the goal is to train a machine learning model on data that is located at different sites and collecting the data in one place is not desired [182]. It requires a collaboration network in which to set up the system and so far only works for machine learning applications. However, if the goal is to share the data freely and not with a dedicated target group (e.g. collaborators) that is known *a priori*, or simply if the data is to be used for explorative analysis and not for the training of a machine learning model, then generating private synthetic data and sharing it instead of the original is the more suitable choice. In this project, we pursued the latter approach, evaluating different generative models for their ability to generate realistic transcriptomic data. Work has been done towards generative modelling in the context of transcriptomics, however, these methods do not ensure privacy [183–186]. It was a collaborative effort with the group from Prof. Dr. Mario Fritz at the CISPA in Saarbrücken, funded by Helmholtz AI under the project title ProGeneGen.

2.2 Methods

2.2.1 Differential Privacy

In order to equip the trained models with rigorous and mathematically sound privacy guarantees, they were trained using differential privacy (DP) [187]. The general idea of DP is that the presence of an individual in a dataset should not significantly increase their privacy risk in comparison to the individual *not* being part of the dataset [188]. DP is built on the concept of ε-indistinguishability. Considering two neighbouring datasets x and x' that only differ in the absence of a single sample in x', these sets are considered ε-indistinguishable if the probability of a query output only varies by a factor of 1+ε between the two sets. The idea is, to add noise to the output, such that ε-indistinguishability is guaranteed for a given ε. The lower the chosen value for the ε-parameter is, the higher are the privacy guarantees of the noised data. At the same time, the level of the noise required to be added becomes greater. A randomised algorithm that provides such indistinguishability of neighbouring sets is referred to as ε-differentially private [189]. A relaxed version of this is (ε, δ)-DP, where the additional δ-parameter provides the probability of breaching the privacy-constraints imposed by DP. In other words, the loss in privacy is bounded by ε with a probability of at least 1-δ, thus, in rare events ε-DP is not guaranteed [189]. The DP parameters chosen for model training were $\varepsilon = 10$ and $\delta = 10^{-5}$.

2.2.2 Evaluation Metrics

To evaluate the quality of the synthetic transcriptomic data, I devised several metrics that reflect different properties of the data.

Classification

As an initial evaluation, our collaboration partners from CISPA assessed the suitability of the synthetic data to train a classifier that then successfully classified the samples of the real data. With each model type, they generated synthetic data first without and then with differential privacy. They then trained a classifier (linear support vector classification) on each of the sets and evaluated its performance on the real data. The model must be able to capture key features of the data for the classifier to train successfully. While this assesses how well the fundamental features were captured, it does not provide any guarantee that the generated data

is biologically meaningful, i.e., that it provides similar analysis results as the original. Therefore, I additionally devised two biology-inspired metrics.

Differential Expression

Differential expression (DE) of genes is a commonly used metric when analysing transcriptomics data of two or more conditions [190–193]. The goal is to identify those genes that are significantly higher or lower expressed in one condition than in another. Such genes are likely to be the underlying drivers of phenotypic differences between the compared conditions. Here, I used a Wilcoxon signed rank test [194] to identify differentially expressed genes. The motivation for choosing this test was its non-parametric character. Many existing methods for DE-gene analysis assume underlying distributions of the gene-expression data [190,193,195], however the true distribution has been a topic of debate for many years [196]. Thus, to avoid making any assumptions to the data that are potentially false, I decided on a test that is non-parametric. For each pair of conditions in the data, I ran a Wilcoxon signed rank test on the measurements for each gene, to identify whether or not the general tendencies of the expression values are different between the conditions. The test was run using the *pairwiseWilcoxon* function from the R-package scran (version 1.26.2)formulating the alternative hypothesis for each side to differentiate between up- and down-regulation. The maximum p-value for a gene to be considered differentially expressed was set to be 0.05.

Gene Co-Expression

Genes that form a functional group, i.e. that are involved in the same biological pathways under a given condition, are often jointly regulated in their expression. Thus, detecting sets of co-expressed genes is a common step in transcriptomics data analysis. To assess whether such functional groups were preserved in the synthetic data, I used *hCoCena* [197], an R-package that my colleagues and I developed and published in 2022, which allows the integration and joint analysis of multiple transcriptomic datasets. This allows us to directly compare gene expression patterns in the real versus the synthetic data in the same analysis. *hCoCena* integrates datasets using a transformation-based approach that utilises gene co-expression networks as the common structure. The tool operates in three phases: 1) The pre-integration phase, where for each transcriptomic dataset the co-expression of each pair of genes is computed. Then, genes that do not show high co-expression with any other gene are removed. Subsequently, a group fold-change (GFC) is calculated for each gene. GFCs are a metric for the average expression of a gene in a group of samples, i.e. all samples of a

particular experimental condition. Lastly, a co-expression network is created for each dataset separately, where nodes represent genes and edges are weighted with the co-expression strength between the two genes they connect; 2) The integration step, where the co-expression networks of the different datasets are integrated into one network. Here I chose the option *integration by union*, where the integrated network consists of all nodes (genes) present in the individual networks, even if some of the nodes are exclusive to a subset of the datasets. This is in contrast to the alternative option *integration by intersection*, where only those nodes are kept, that are present in the individual networks of *all* datasets. I chose the former option since it allows a more holistic analysis of the data at hand, also incorporating and visualising the genes that the generative model failed to properly generate. A high amount of edges can be expected to exist in multiple individual networks, however with different weights. During the integration step, I thus chose to keep the lowest weight in case of multi-edges. The reasoning here was that this represents a sort of lower boundary, not assuming a higher expression than the true expression value. Additionally it prevents the integrated network from becoming too dense with false positive co-expressions, hindering subsequent community detection in the network; 3) In the final post-integration phase, the integrated network is analysed. I selected the *leiden*-community detection algorithm to detect communities - or clusters - of densely connected nodes in the integrated network. I then visualised the mean GFC of those clusters across the different conditions for both real and synthetic data. If the generative model recognizes patterns of co-expression and functional modules of genes and is capable of reflecting that biology in the synthetic data that it generates, then the mean GFCs across detected clusters will reflect that.

2.2.3 Generative Models

Several types of generative models were assessed for the purpose of generating synthetic transcriptomic data by our collaboration partners.

DP-WGAN

The DP-WGAN is a Wasserstein-GAN (WGAN) [198] trained with differentially private stochastic gradient descent (DP-SGD) [199]. WGANs are GANs that are trained using the Wasserstein distance. The Wasserstein distance of the true and generated data distribution is smoother than the Kullback-Leibler or Jensen-Shannon divergence used in “classical” GANs and therefore allows more stable gradient descent. If the generative process requires

differential privacy, DP-SGD can be used to noise and clip the gradients during training such that a given privacy guarantee can be achieved.

DP-CVAE

The Differentially Private Conditional Variational Autoencoder (DP-CVAE) is a conditional autoencoder that was trained with differentially private stochastic gradient descent to enable private data generation. The KL-term of the DP-CVAE loss was weighted with $\beta = 0.001$.

DP-MERF

Differentially Private Mean Embeddings with Random Features (DP-MERF) [200] is a method that enables the training of a private data generator by evaluating its performance based on the Maximum Mean Discrepancy (MMD) between kernel mean embeddings of the real data and that of the synthetic data. Privacy is introduced by perturbing the mean embedding of the real data set. This has to be performed only once and can then be re-used throughout the training of the generator.

2.3 Dataset

A bulk-RNA sequencing dataset with 1181 samples spanning various diseases was used which was compiled by Warnat-Herresthal *et al.* [201]. For this use case, we focused on generating different types of leukaemia samples versus samples from the category “other”, comprising a wide variety of controls and other diseases. The leukaemia types present in the dataset were acute myeloid leukaemia (AML), acute lymphocytic leukaemia (ALL), chronic myeloid leukaemia (CML) and chronic lymphocytic leukaemia (CLL). The “other” category accommodates samples from influenza vaccination, Lyme disease, Plasmodium vivax challenge, juvenile idiopathic arthritis, polycystic kidney disease, myelodysplastic syndrome, tuberculosis vaccination and other conditions with low sample counts. The classes used for training were unbalanced, posing an additional, but realistic challenge for the generative model. The sample counts per condition are listed in Table 1.

Condition	AML	ALL	CML	CLL	Other
# Samples	508	12	14	13	634

Table 1: Samples per condition. Number of samples for each condition in the real transcriptomics dataset. AML: acute myeloid leukaemia, ALL: acute lymphocytic leukaemia, CML: chronic myeloid leukaemia, CLL: chronic lymphocytic leukaemia. "Other" comprises a variety of different conditions (no leukaemia) and controls.

According to the publishers of the dataset, the RNA-sequencing data has been normalised using DeSeq2 [195]. For training the generative models, the data was standardised by subtracting the mean and dividing by the standard deviation of each feature (i.e. gene) and filtered for a subset of approximately 1,000 genes. The filtering step was motivated by the "curse of dimensionality", since the number of features (genes) was more than 10 times higher than the number of samples. The subset of genes was not selected randomly but based on so-called *landmark genes* identified in the LINCS L1000 project [202]. These genes were found to be representative of approximately 20,000 other genes that could be inferred by only measuring the expression strength of the landmark genes. Given this information redundancy in the data, we subset the dataset to only these landmark genes.

2.4 Results

Choosing a Model

Training the DP-WGAN on the transcriptomics data was very unstable and training a classifier on the generated data to classify the original data also showed poor performance with a classification accuracy of only 54.1%. Thus, this model type was excluded for the task at hand. DP-CVAE training was much more stable and yielded classifier accuracies of above 93% for both, non-private and private generation (Table 2), while DP-MERF generated data that allowed good classification performance in the non-private setting but poor performance in the private setting. Thus, the conditional variational autoencoder was chosen as the most promising model. The generative models and classifiers were trained and evaluated by our collaboration partners from CISPA.

	non-private	private ($\varepsilon = 10$)
real data	99.2%	-
DP-MERF	94.5%	50.2%
DP-VAE	99.2%	93.2%

Table 2: Classifier accuracies on synthetic data. Shown are the classifier accuracies achieved when training a classifier on synthetic data generated by the listed models and testing its classification performance on the real data. Accuracy is

measured for training the classifier on non-private synthetic data (left column) and on differentially private synthetic data (right column).

Differential Expression

I performed the differential expression analysis to detect DE-genes for all condition-comparisons and compared the results yielded in the real, the synthetic and the synthetic data with DP. The correctly reconstructed DE genes in the synthetic data and the synthetic data from the DP model aggregated over all condition-comparisons are illustrated in figure 17. As shown, the percentage of correctly reconstructed DE-genes in both the up- and the down-regulated genes is high for the synthetic data without DP. For the synthetic data with DP, however, the correct DE-genes are much lower with a median only around 25% for up-and downregulated genes. Supplementary figure 1 shows the DE-gene preservation in more detail for each of the condition-comparisons and the different datasets. The overlap between the DE-genes across comparisons for the real and the synthetic data without DP is generally high, which is not the case for the comparison between real and synthetic data with DP, with larger proportions of the DE-gene sets not shared between the real and the synthetic data with DP.

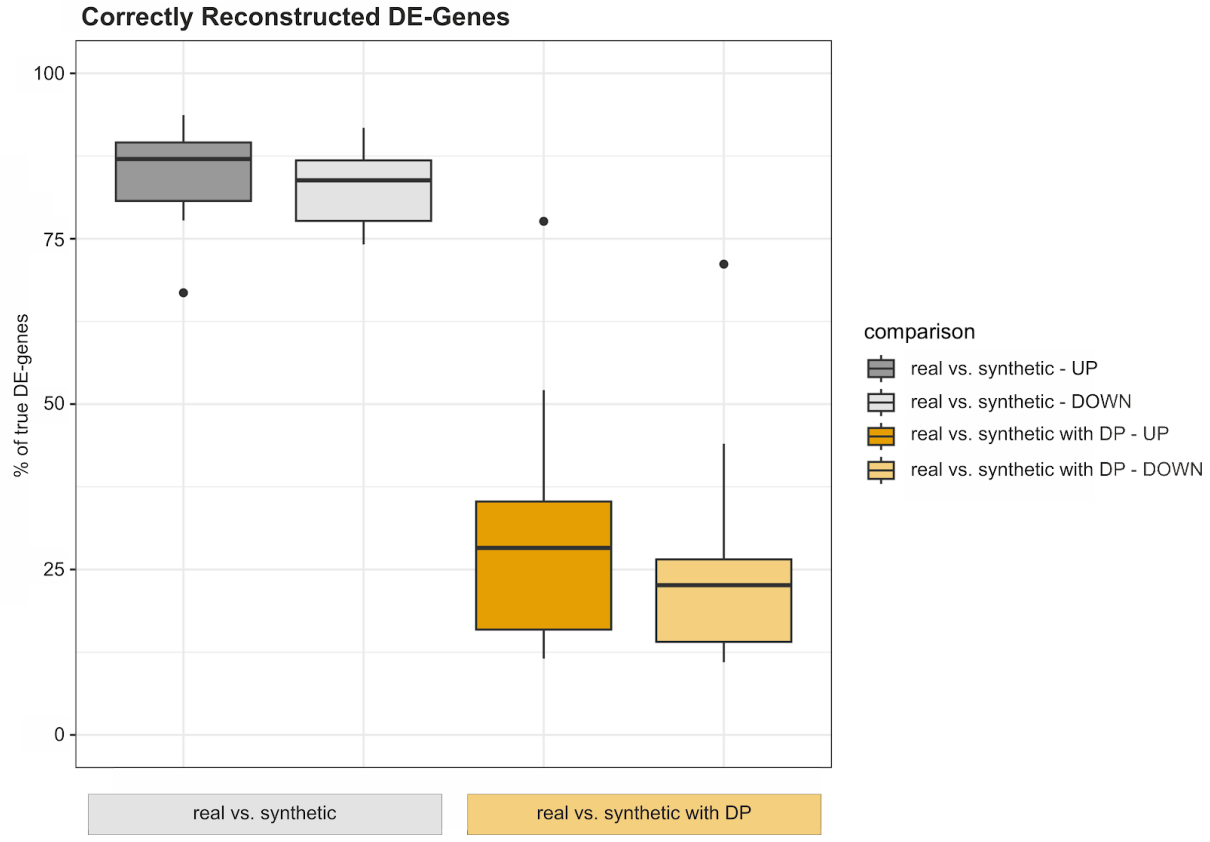


Figure 17: Differential expression. Box plot illustrating the percentage of correctly reconstructed differentially expressed (DE) genes when comparing the real data with the non-private synthetic data (grey) and the private synthetic data (orange).

Darker shaded box plots represent upregulated DE-genes and lighter shaded ones represent down-regulated DE-genes. DP: differential privacy.

Gene Co-Expression

As described above, in an ideal scenario, the co-expressions of the genes present in the dataset should be maintained across the synthetic samples. However, perfect preservation seems unfeasible, therefore in a first step, a reasonable baseline had to be found in order to measure "good" performance. I computed the preserved co-expressions in the test vs. the train split of the real data. The motivation here was that all samples come from the same dataset and thus from the same underlying distribution. The preservation of co-expression between test and train split therefore serves as a good baseline. I computed the pairwise correlations for all genes in the train and the test split. These gene paris were then filtered using 100 equally spaced correlation-cutoff values between 0 and 1. As shown in the top panel of figure 18, even when comparing the train and the test split, 100% preservation of co-expression are not preserved, validating that this ideal scenario indeed would not serve as a realistic base-line.

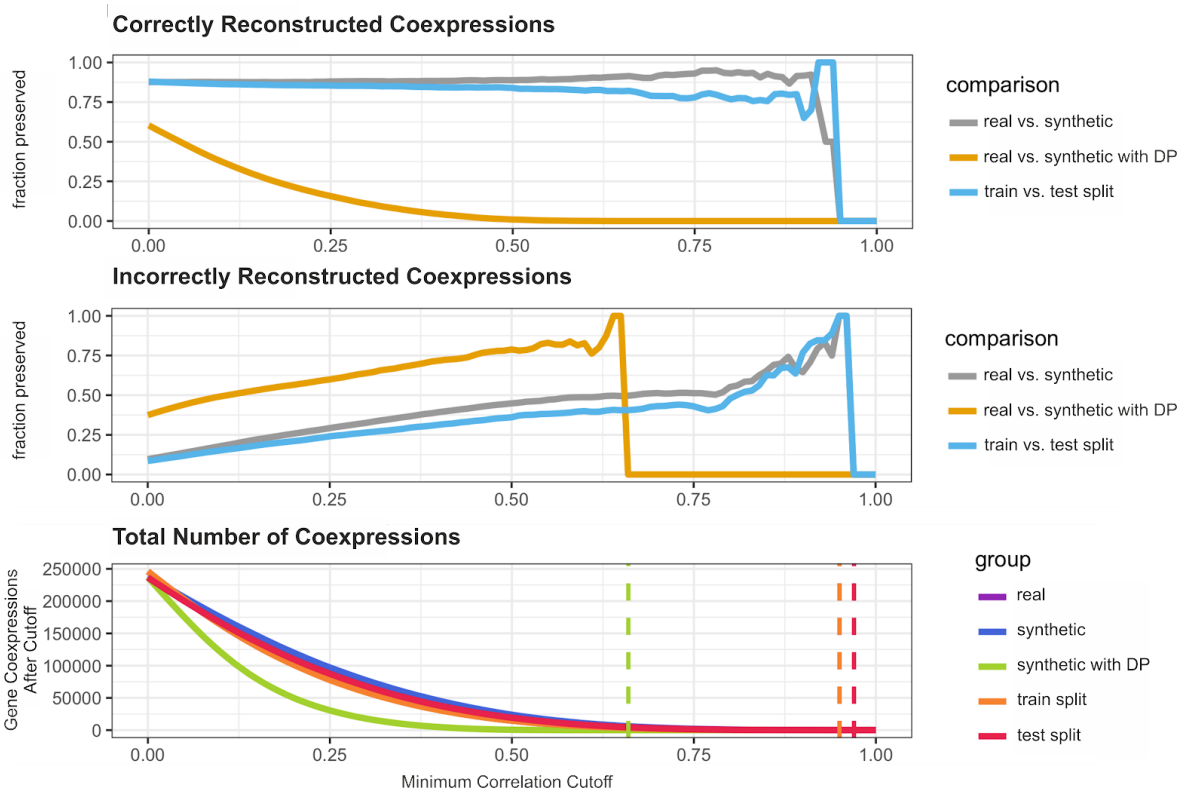


Figure 18: Preserved co-expressions across different correlation cutoffs. The x-axis shows the Pearson correlation coefficient used as a cutoff to filter for co-expressions that exceed it. The y-axis shows the percentage of co-expressions from the reference (the real set or the train set) that are preserved in the synthetic, synthetic with DP or test set after applying a cutoff. **Top**: The percentage of co-expressions that are correctly preserved in the synthetic, private synthetic or test data. The train vs. test comparison serves as a baseline (blue). **Middle**: The percentage of co-expressions in the synthetic, private

synthetic or test data that are incorrect, i.e. that do not exist in the real/train data. The train vs. test comparison serves as a baseline (blue). **Bottom**: The total number of co-expressions in the different datasets that exceed the illustrated correlation cut-offs. Dashed lines in corresponding colours indicate the cut-off at which the dataset has no co-expressions left. The purple line (real data) is not visible because it is overlapped by the orange one (train data), the blue line (synthetic data) is not visible because it is overlapped by the red one (test data).

Generally, the preservation of co-expressions between real and non-private synthetic data (i.e. without DP) and train vs. test split are similar, indicating good reconstruction performance of this metric in the non-private synthetic gene-expression data. However, the preservation of these co-expressions in the private synthetic data (i.e. with DP) is much lower, dropping rapidly to close to zero. At the same time, as illustrated in the middle panel, the fraction of co-expressions that are incorrect, i.e. that exist in the synthetic data but not in the real data, is very high in the private synthetic data and much lower (and close to the values in the train vs. test scenario) in the non-private synthetic data. The sudden drop to zero in the fraction of incorrect co-expression in the comparison real vs. private synthetic is due to the fact that after applying that cutoff there are no more edges left in the private synthetic data (green dashed vertical line in bottom panel). Generally, it can be observed in the bottom panel, that the total number of co-expressions that exceed each cutoff are lower in the private synthetic data, while all other sets, including the non-private synthetic data, are similar.

Next, I applied *hcocena* to the real and the non-private synthetic transcriptomic data. One sample was removed from the real dataset due to outlier characteristics, a sample from the "other" category (suppl. figure 2). PCAs of each dataset using the data from all present genes revealed a branching in the "other" category in the synthetic data that cannot be observed in the real data (Figure 19 A, B). The correlation cut-off was then set to be 0.7 to only include strong co-expressions that are potentially biologically meaningful. After application of this cut-off and filtering the co-expression values for p-values < 0.05, 354 nodes and 2035 edges were left for the real data and 508 nodes as well as 3893 edges for the synthetic data. After network integration, clusters of co-expression were detected using the *leiden* algorithm and mean GFCs were computed. As illustrated in Figure 19 C, for each condition the real and synthetic samples cluster together and show highly similar mean GFCs across the detected clusters. The clusters detected in the integrated network show distinct spatial arrangement (Figure 19 D). Plotting the network for each condition and each dataset with nodes coloured by GFC per gene emphasises the similarities in expression pattern per gene between the real and the synthetic data (suppl. figure 3).

Running the *hcocena* analysis for all threes sets (real, non-private synthetic and private synthetic) illustrated again, that the real and the non-private synthetic data exhibit very similar expression patterns, whereas the private synthetic data behaves differently, as indicated by the private data clustering far away in the heatmap across conditions (suppl. figure 4). Note, that for this analysis, the correlation cut-off had to be set to a lower value (0.5), because there were barely any co-expressions in the private synthetic data that reached high values.

To investigate whether the biologically high similarity between the real data and the non-private synthetic data is due to the model copying from the real data, a nearest neighbour analysis was performed based on euclidean distance. It revealed that for the vast majority of non-private synthetic samples (78%), the nearest neighbour was another synthetic sample and only in 22% of the cases was the nearest neighbour a real sample (Figure 19 E). Inversely, when identifying the nearest neighbours of the real samples, the nearest neighbour of 49% of all real samples was another real sample and 51% of the real samples had a synthetic sample as their nearest neighbour.

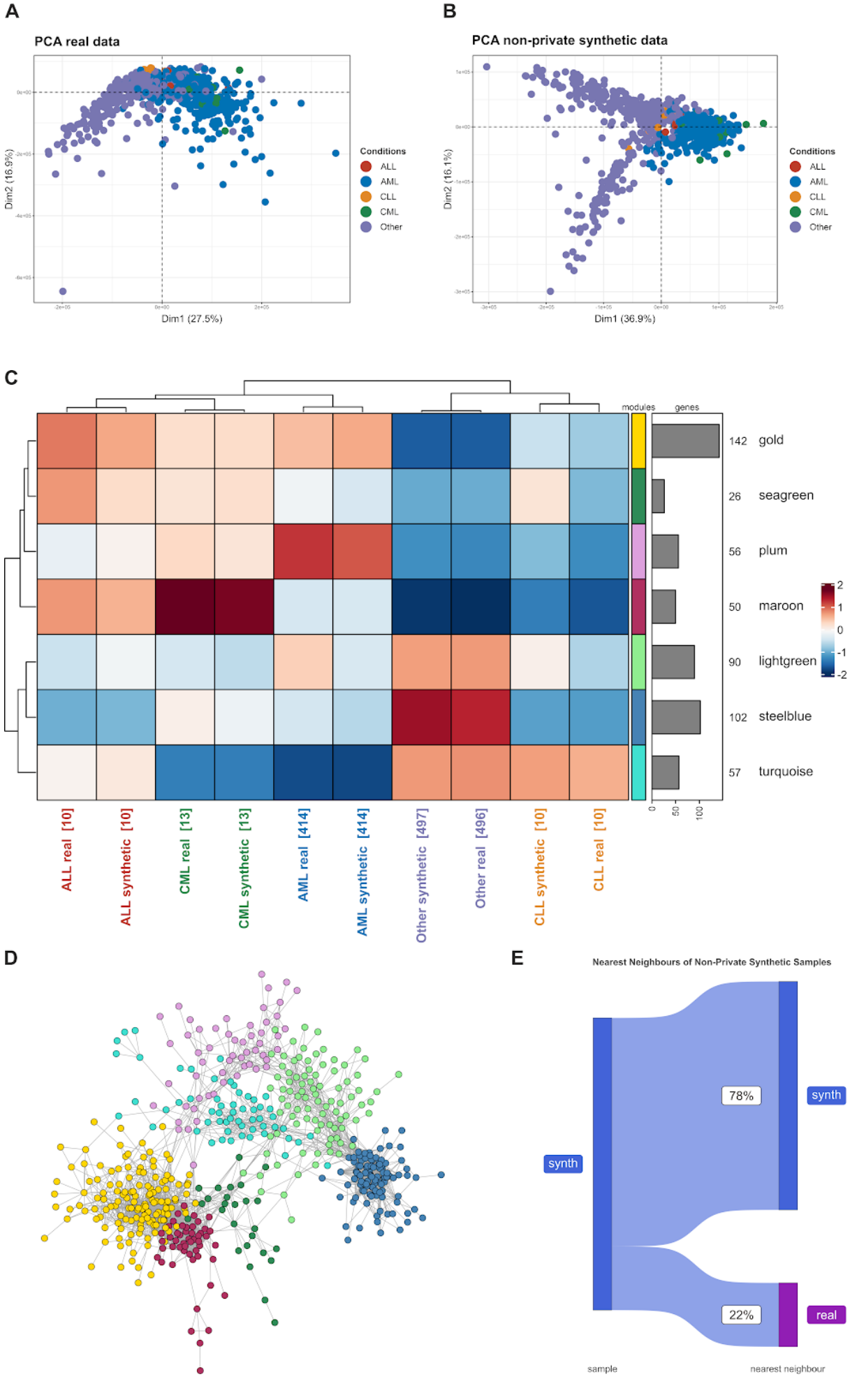

A
PCA real data
Dim2 (16.9%)
Dim1 (27.5%)
Conditions
ALL
AML
CLL
CML
Other
B
PCA non-private synthetic data
Dim2 (16.1%)
Dim1 (36.9%)
Conditions
ALL
AML
CLL
CML
Other
C
modules
genes
142 gold
26 seagreen
56 plum
50 maroon
90 lightgreen
102 steelblue
57 turquoise
2
1
0
-1
-2
0
50
100
ALL real [10]
ALL synthetic [10]
CML real [13]
CML synthetic [13]
AML real [414]
AML synthetic [414]
Other synthetic [497]
Other real [496]
CLL synthetic [10]
CLL real [10]
D
E
Nearest Neighbours of Non-Private Synthetic Samples
synth
78%
synth
22%
real
sample
nearest neighbour

Figure 19: *hcocena* results and nearest neighbour analysis. **A**) PCA of the samples present in the real data. Samples are coloured according to condition; **B**) PCA of the samples present in the non-private synthetic data; **C**) Heatmap showing the mean GFC for each gene-cluster across conditions. Conditions are clustered based on their mean GFC patterns using hierarchical clustering. Barplots on the right indicate the number of genes present in each cluster; **D**) The integrated gene co-expression network with nodes (genes) coloured by the co-expression cluster they belong to; **E**) Sankey plot of the nearest neighbours of the synthetic samples.

2.5 Discussion

The goal of this project was to assess the potential of generative models in the context of data privacy of high-dimensional medical data spaces, exemplified on transcriptomics data. The generated data was evaluated for different levels of utility. The results demonstrated that both non-private and private synthetic data allowed successful training of a high accuracy classifier to classify real samples. However, the following evaluations for biological utility revealed that the non-private synthetic data offers high utility, while the private data loses its utility. Differential expression of genes was maintained to a large proportion in the non-private data but poorly preserved in the private data. A similar picture was painted in terms of preserved co-expressions between genes. Here, the private data not only underperformed in their preservation but also introduced a high number of false co-expressions and generally lower co-expression strengths between genes. The non-private data on the other hand showed a similar conservation of true co-expressions as the test data to the train split. The discrepancy between the co-expression in the train vs. the test split to the hypothetical ideal baseline of 100% preservation is likely due to the data being noisy. Sources for this noise are likely two-fold: biological and technical. The technical noise can originate from various sources such as sample handling, personnel, sample storage time or technology used.

The *hcocena* analysis emphasised the high biological utility of the non-private synthetic data. The branching of samples in the category “other” that was observed in the PCA of non-private synthetic samples might be induced by the extreme heterogeneity of this category, comprising a large number of different diseases. A reason for the branching could thus be an overpopulation of certain diseases in the “other” category that are minority samples in the real data, such as the removed outlier sample. The fact that the cluster heatmap clusters by condition rather than real vs. non-private synthetic data indicates that the generative model was able to detect and reconstruct functional groups and their expression patterns. The high utility and the absence of privacy constraints in the non-private synthetic

data introduced the concern that the model's generalisation could be poor, i.e. that the model mainly copies samples from the original data. In such a case, a high proportion of the synthetic samples would be expected to have a real sample as their nearest neighbour (excluding the sample itself). However, the nearest neighbour analysis showed that the nearest neighbour of most synthetic samples was another synthetic sample. While this points towards good generalisation, the fact that the synthetic data is more similar to itself than it is to the real data also indicates that there exists a batch effect introduced by the generation that makes the real and the synthetic data distinct. Future investigations of the exact nature of this effect are needed. Generally, the biological evaluation revealed that the private synthetic data has only minimal biological utility. While it protects the privacy of patients participating in the underlying studies, important gene-expression trends that allow insight into the underlying conditions are lost and it seems that the trade-off between privacy and utility is too large. While increasing the bound in the privacy loss (ε) would reduce the privacy guarantees, future experiments should include gradually increasing this parameter to identify a potential tipping point below which the data loses its utility. Including biological utility, e.g. the preservation of DE-genes or co-expressions, into the model training could improve the synthetic data, however, the incorporation of such data characteristics would have to respect the privacy budget to not undermine the privacy guarantees.

All together, the generative model presented here demonstrated that highly complex, non-normally distributed data such as gene expression data can be learned and generated, but making this process private still requires additional research to maintain utility. This includes the consideration to incorporate biology-based metrics into the training process. However, here a limiting factor is that the incorporated information must also be considered under the given privacy constraints. Future directions should also include the private generation of single-cell RNA-seq data, which is increasingly used in the biomedical field. However such data differs strongly from bulk-RNA-seq data, especially due the high number of missing values and the dependencies between samples, more concretely between cells of the same individual. In this case, the entity to be protected with DP would not be a sample (as is the case in bulk-data), but instead groups of samples, i.e. all cells belonging to the same individual.

3. Systematic Evaluation of Autoencoder Architectures on the Encoding Quality of Molecules

3.1 Introduction

After collecting transcriptomic data from a patient cohort it is not only essential to share the data with the research community but also to use it to find potential cures for the medical conditions from which the patients suffer. Identifying therapeutic drugs, be they existing ones or novel ones, can benefit from systems medicine approaches like transcriptome sequencing. However, as a very first step in systems medicine-based drug development the molecules have to be made machine readable to process them computationally. A common method is to embed the molecules into a numerical vector of fixed length using (Variational) Autoencoders. Many different (V)AEs are described in literature for embedding molecules [19,21,78,171,203–205], however their design seems mostly arbitrary and there is often no assessment of the latent space quality before using it for downstream tasks. A systematic approach to better understand these aspects is necessary, especially because the quality of downstream applications, e.g. the training of machine learning models, is highly dependent on the quality of the embedding. However, understanding how architecture impacts model performance is not only important to ensure quality, but also to optimise the training procedure for a given learning task. Optimised training is a particularly interesting aspect in the context of green computing and the democratisation of AI: Training AI models that beat state-of-the-art performances demands increasingly specialised hardware and often takes weeks to train. Such training conditions are incredibly expensive, making them exclusively available to large companies and wealthy countries. Additionally, their carbon footprint is alarmingly high: Strubell *et al.* [23] estimated that the CO_2 emissions of training a large transformer network with neural architecture search are approximately 600k pounds, which is 5-times as much as the emissions of an average car, including fuel, over its entire lifespan. Therefore, researchers are demanding solutions for greener and more publicly available AI [22,23]. Task-specific engineering of models to reduce model size, the size of the required training data as well as training time can reduce energy consumption and lower hardware specificity, making the models more accessible and environmentally friendly.

Thus, in this project, I assessed how different architectural choices as well as molecule input types affect the utility of the latent space generated by Autoencoders, how this compares to the latent space of Variational Autoencoders and in what way these insights can be utilised to optimise the model architecture for high-quality embeddings while reducing the resources spent on training.

The success of the models in meaningfully encoding the provided molecules is evaluated based on reconstruction metrics, information content of the latent space as well as the chemical knowledge represented in it.
Initial code development and experiments on SMILES were done by our Bachelor student Iva Ewert over the course of her thesis and under Dr. Matthias Becker's and my supervision. Afterwards, I expanded the code, experiments and analyses to an extent that the results presented here are a significant extension that goes beyond the work of her thesis.

3.2 Methods

3.2.1 Models

I compared models with SMILES as well as models with SELFIES as input. Since both molecule representations are text-based and of differing length, I used Recurrent Neural Networks (RNNs) for the encoder and decoder structure. More specifically, I tested both Gated Recurrent Units (GRUs) [73] and Long Short-Term Memory (LSTMs) [69] architectures. I defined a base-model for both architecture types, the performance of which was used as a point of reference. I then separately adjusted different parameters and evaluated their impact on performance: The hidden and the latent sizes of the RNN cells, the number of layers in decoder and encoder as well as the use of the attention mechanism. The different architectures investigated in these experiments are listed in Table 3.

experiment	model type	latent size	hidden size	layers	attention
base model	GRU	64	64	1	no
base model	LSTM	64	64	1	no
hidden + latent	GRU	↓ **32**	↓ **32**	1	no
hidden + latent	LSTM	↓ **32**	↓ **32**	1	no
hidden + latent	GRU	↑ **128**	↑ **128**	1	no
hidden + latent	LSTM	↑ **128**	↑ **128**	1	no
latent	GRU	↓ **16**	64	1	no
latent	LSTM	↓ **16**	64	1	no
latent	GRU	↓ **32**	64	1	no
latent	LSTM	↓ **32**	64	1	no
layers	GRU	64	64	↑ **2**	no
layers	LSTM	64	64	↑ **2**	no
layers	GRU	64	64	↑ **3**	no
layers	LSTM	64	64	↑ **3**	no
attention	GRU	64	64	1	**yes**
attention	LSTM	64	64	1	**yes**

Table 3: Overview of the architectural changes made in each experiment. In bold are architecture features that were modified in comparison to the base models. Arrows indicate if the feature was increased or decreased in comparison to the baseline.

The base models were a GRU-based as well as an LSTM-based architecture with a latent and hidden size of 64, 1 layer and no attention. Latent and hidden size were chosen to be 64 because it exceeds the maximum sequence length for both SMILES and SELFIES in the dataset. Such a latent size would thus not force the model to perform extensive data reduction and allow us to experimentally test the impacts of modifying this parameter. More details on lengths and token distributions are provided below, under *Datasets*. The models were trained with at least 50 epochs. Then, training was stopped by one of two events, depending on which occurred first: the maximum of 500 training epochs was reached or early stopping took effect due to no further improvement. Early stopping was configured to take place if after 5 validation rounds the validation loss had not improved by at least 0.01. An Adam optimizer with learning rate 0.005 was used. Additionally, teacher forcing [206] was used to avoid accumulative error after predicting a wrong token. This is a common issue faced particularly in early training steps, since in those stages, the model makes many errors. An incorrectly predicted token at the beginning of the sequence would cause all following token predictions to build up on the error, if the incorrectly predicted token was used as an input to the next step. To avoid this, the true token is provided to the next prediction step rather than the (potentially wrong) predicted token. The loss that was used in these experiments is a cross

entropy loss acting on the predicted token for each position in the string versus the true token per position. Every model was trained using three different seeds, these seeds were the same across all experiments. Results are reported as mean values across those three runs.
To indicate the beginning and the end of a sequence, the molecules were padded with a start and a stop token and if necessary padding tokens were added, if the sequence was shorter than the maximum sequence, such that the training could be performed in mini batches. The molecules were then one-hot encoded before passing them as input and mini batch size was set to 256. In case of GRUs, the hidden state after processing the last input token was used as the latent representation. Since LSTMs return both their hidden as well as their cell state, the two states were concatenated and passed to an additional linear layer to create the latent representation. This additional linear layer was necessary given the otherwise double size of the concatenated tensors. Further, the decoder was initialised using the latent representation as its hidden state in case of GRUs, for LSTMs, two separate linear layers were used to reconstruct the cell state and the hidden state from the latent representation. These were subsequently used to initialise the hidden and cell state of the LSTM-decoder.

3.2.2 Evaluation Metrics

The reconstruction performance of the models was assessed using two metrics: mean similarity and full reconstruction. The mean similarity is the percentage of correctly reconstructed tokens per molecule, averaged across all molecules. The full reconstruction rate is the percentage of all molecules that were reconstructed correctly in their entirety. If any token was incorrect, the molecule was considered wrong. Thus, the full reconstruction rate is a much harder metric than the mean similarity because it requires the model to also learn and correctly reconstruct rare tokens for this metric to be high.

3.2.3 Latent Space Analysis

If the molecule encodings are meant to be used for downstream tasks such as training another machine learning model, then good reconstruction of the input molecules is not sufficient. The latent space must also be organised in a chemically meaningful way. Molecules that are similar, be that in structure or in function - although these two often (but not always) go hand in hand - should be placed in close proximity in the latent space, whereas different molecules

are expected to be further apart. Depending on the molecule representation used for training the model, different approaches can be taken. SMILES for example have the characteristic that they are not unique, unlike SELFIES. While this redundancy is often a hindrance, in this case it is an opportunity: The same molecule, independent of the exact SMILES that represents it, should have the same or at least a very similar latent representation. Thus, to evaluate the latent space of the models trained on SMILES, three molecules were picked at random from the test set. For each of these molecules, 5 enumerations, i.e. different SMILES that describe the same molecules, were generated. The enumerated molecules were then embedded using the model's encoder. Two tests were performed on their embeddings: First, the average Euclidean distance between the embeddings of each molecule's enumerations was calculated. As a point of reference, 1,000 molecules were sampled at random, embedded and their pairwise distances were calculated as well. The expectation is that in a chemically meaningful latent space the embeddings of different enumerations of the same molecule are more similar than the embeddings of random molecules. In the second test a Principle Component Analysis (PCA) was computed based on the enumerated embeddings as well as the embeddings of 10,000 randomly selected molecules. Their spatial arrangement was then visualised using the first and second principle components. Enumerations of the same molecule are expected to cluster tightly together.

SELFIES, however, are unique for each molecule and thus the option of enumerating them is not given. But since SELFIES guarantee validity, they offer the option of introducing point mutations, i.e. randomly exchanging a single token in the SELFIES sequence. Thus, again three molecules were randomly chosen from the test set. For each of the three molecules, a single point mutation was introduced 20 times and the four mutated molecules with the highest Tanimoto similarity [207] to the original molecule were selected, yielding in total five representations for each molecule (including the original representation). They were then evaluated in the same way described above for the SMILES models: their Euclidean distances were compared to those of other random molecules and their spatial localisation in a PCA was visually inspected.

3.3 Datasets

I used the Molecular Sets (MOSES) dataset to train the models. MOSES is a benchmarking dataset developed for comparable and standardised training of machine learning models [208]. It is a subset of the Zinc Clean Leads [209] dataset that contains more than 4.5 million

molecules. These molecules are characterised by having a molecular weight of at least 250 Da and at most 350 Da, no more than 7 rotatable bonds and an XlogP [210] (a way to calculate the partition coefficient of molecules in water and octanol) of at most 3.5. To create the MOSES set, the molecules were filtered for molecules without charged atoms, without atoms other than C, N, S, O, F, Cl, Br and H, as well as no cycles that have more than eight atoms. They further applied so-called medicinal chemistry filters. These are filters that are meant to exclude molecules that are likely to form toxic intermediates or are toxic metabolites themselves. Additionally, they applied PAINS (pan-assay interfering compounds)-filters, to remove molecules that are likely to show up as false positives in assays and that are predicted to have unspecific bioactivity. The final MOSES set comprises 1,584,664 training molecules and 176,075 test molecules. I created an additional train set of 50,000 and a test set of 12,500 molecules by randomly sampling the MOSES set. The motivation for this was to assess the capability of the models when they were trained with a significantly smaller dataset, especially encouraged by prospects of green computing. The token and string length distributions were very similar in the full set and the subset (Figure 20), indicating that the subset is representative of the full set.

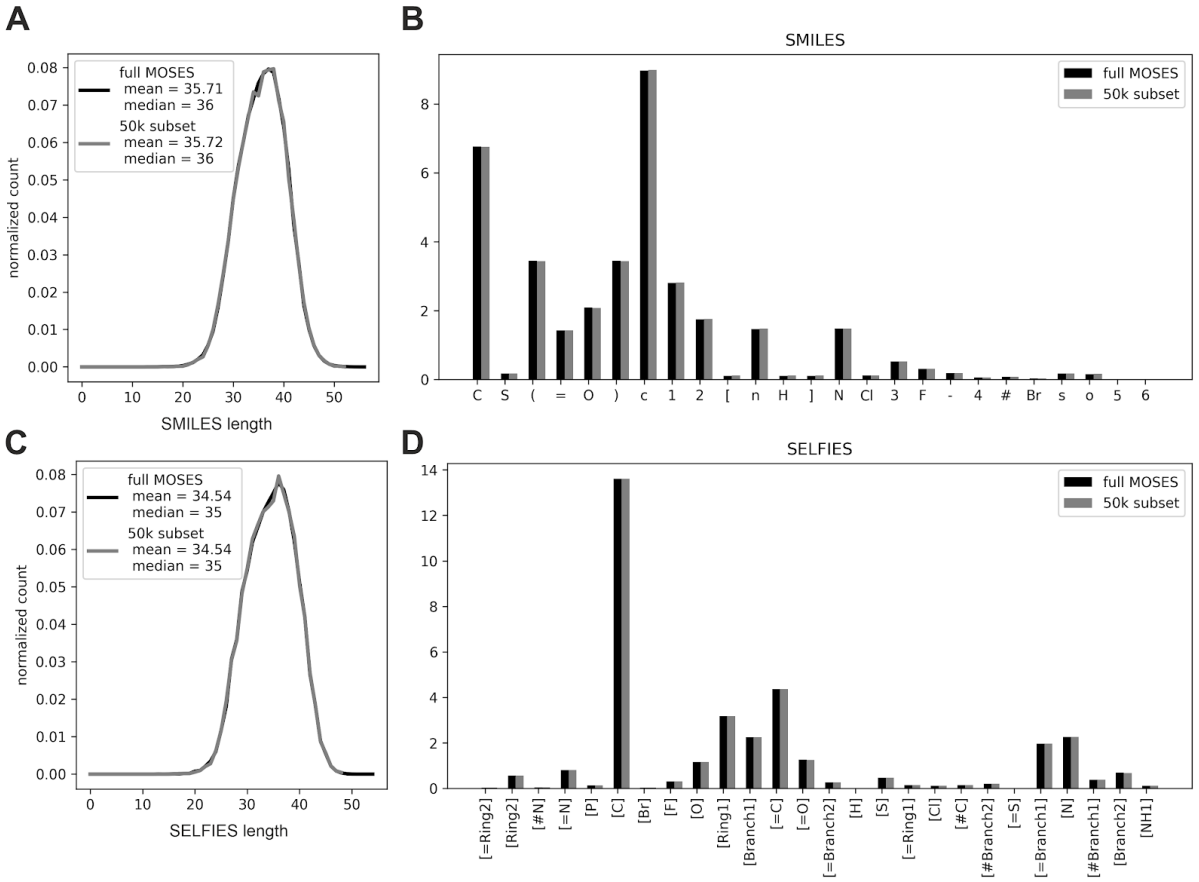


Figure 20: String lengths and token distributions in the MOSES set and the subset. **A**) Histogram of the SMILES string lengths in the full MOSES set (black) versus the 50k subset (grey). Counts were normalised to the number of molecules present in the set. **B**) Token frequencies of SMILES tokens in the full set versus the subset. **C**) Same as **A** but for the SELFIES translations of the SMILES. **D**) Same as **B** but for the SELFIES translations of the SMILES.

3.4 Results

Baseline Models

When evaluating the reconstruction performances for both, the model using SMILES as input and the model using SELFIES as input, the GRU-based architecture outperformed the LSTM (Figure 21). This observation held for the 50k subset as well as the full MOSES dataset. As expected, since it is the much simpler metric, the mean similarity was generally higher than the full reconstruction. Mean similarity for GRU in the 50k subset was on average 83.8% (SELFIES) and 84.8% (SMILES) and improved further in the full set to 96.9% (SELFIES) and 97.6% (SMILES). For the LSTM architecture, it improved from 74.8% (SELFIES) and 75.4% (SMILES) in the subset to 85.6% (SELFIES) and 94.3% (SMILES) in the full set. Generally, models trained on SMILES performed better than those trained on SELFIES, however mostly only by a small margin, except for the LSTM architecture on the full set where the performance of SMILES was noticeably better.

The full reconstruction was extremely poor in the 50k subset for both molecule representations and across RNN architectures. In fact, the LSTM fails to fully reconstruct any of the molecules. While this is much improved for the GRU when using the full set, elevating the metric score by more than 48% for both input types, increasing the set size only slightly improved the performance of the LSTM: full reconstruction of SMILES increases from 0% to 5.7%, but remains unchanged at 0% for SELFIES.

The performance of these baseline models will be used throughout the following experiments to assess how model performance changes when making different architectural decisions.

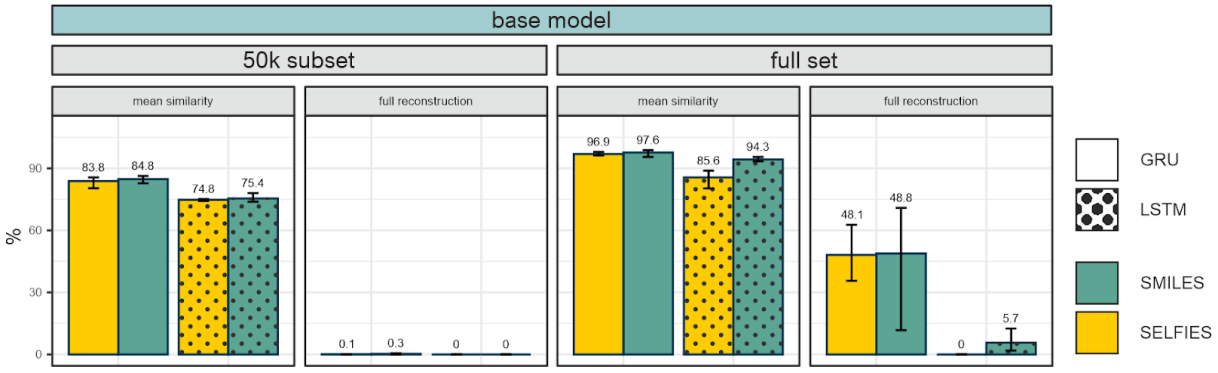


Figure 21: Reconstruction performance of the base model. Shown are the mean similarity and the full reconstruction of SMILES (green) and SELFIES (yellow) for the 50k subset and the full MOSES set. GRU models are shown with a solid fill, LSTM models have a dotted fill. Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

Jointly Changing Hidden and Latent Size

In this experiment, both the hidden size and the latent size were a) decreased to 32 and b) increased to 128 and the performance in both cases was compared to the baseline model with hidden and latent size of 64 (Figure 22). For the full dataset, the mean similarity decreases distinctly for the GRU-architecture when reducing the sizes to 32, dropping from 96.9% (SELFIES) and 97.6% (SMILES) to 80.5% (SELFIES) and 83.7% (SMILES). Given the already high scores in the baseline model, increasing the sizes 128 only has small effects, increasing the mean similarity to 99.5% for both molecule representations. Overall, for the GRU trained on the full set, a clear trend of larger layers to higher similarity can be observed. Looking at the LSTM architecture, the results are not as clear-cut. Firstly, there is a more distinct difference between SMILES and SELFIES than can be observed in the GRU. SMILES strongly outperform SELFIES in this architecture, particularly for the largest model with sizes 128, where SMILES reach a mean similarity of 94.5% in the LSTM, while SELFIES show approximately 15% worse performance. Additionally, the SMILES-results in the LSTM architecture are more stable, albeit lower, across the different sizes than they are in the GRU: with 91.2%, the LSTM trained on SMILES at size 32 is almost 8% better than the GRU counterpart. However, with increasing sizes, this does not improve much: 94.3% and 94.5% for sizes 64 (baseline) and 128, respectively. Interesting are the observations for LSTMs with SELFIES. Here, both reducing and increasing the hidden and latent size have unfavourable effects and reduce the mean similarity from 85,6% in the baseline to 77.7% and 79.8% for size 32 and 128, respectively. In the 50k subset, the picture is a bit clearer. For both the GRU and the LSTM, increasing the sizes improves mean similarity. GRUs generally perform better than the LSTMs and the performance on SELFIES and SMILES is comparable.

In terms of full reconstruction performance, increasing the hidden and latent sizes to 128 made quite the difference for the GRU-architecture in the 50k subset. Here, the base model and - expectedly - also the smaller model of size 32, are not able to fully reconstruct (any) input molecules. However, increasing the sizes to 128 gave a boost of almost 9% in case of SELFIES and 6.6% for SMILES. For the LSTM architecture, the full reconstruction remained unchanged at 0%. Training on the full MOSES dataset, the full reconstruction rate

for GRUs almost dropped to zero when decreasing the sizes to 32. Increasing them to 128, however, elevated the full reconstruction to 90.7% and 86.4% for SELFIES and SMILES, respectively. In LSTMs, full reconstruction remained at zero across sizes when using SELFIES as input. When using SMILES, models with sizes 32 and 64 performed similarly (approximately 6%), but increasing to 128 boosted full reconstruction to an average of 28.1%. Notably, the three different seeds demonstrated a high variation in this case, one giving full reconstruction results of almost 80%.

Overall, the results yielded when changing both latent and hidden size were mixed and strongly dependent on molecular input type, RNN architecture and dataset. The reasons for the particularly unstable performance of LSTMs can only be speculated about. One potential cause might be their much larger number of trainable parameters in comparison to GRUs, owed to their higher number of gates. Increasing the hidden and latent sizes further amplifies this, potentially leading to too many trainable parameters.

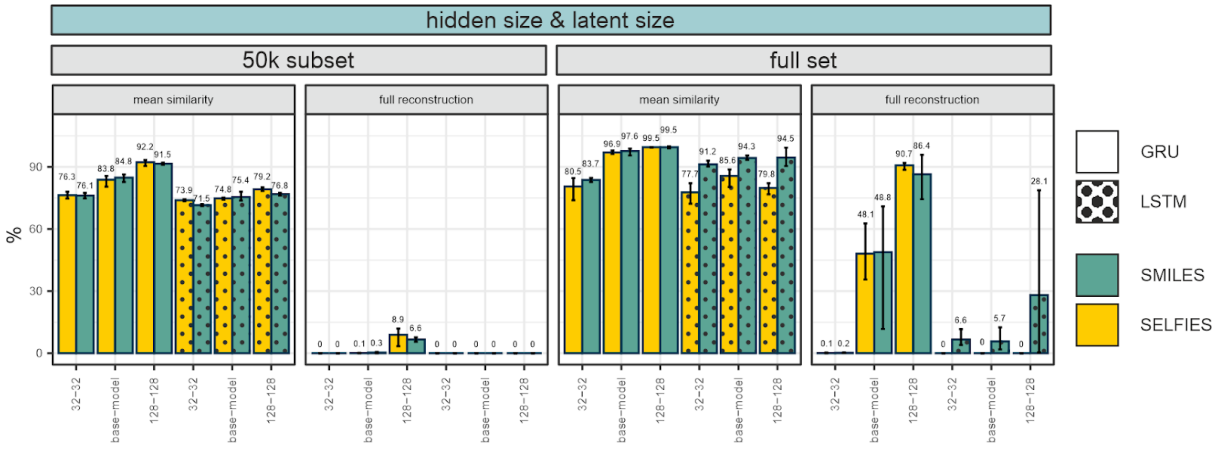


Figure 22: Reconstruction performance of models with manipulated latent and hidden size. Results are shown in comparison to the baseline models. Shown are the mean similarity and the full reconstruction of SMILES (green) and SELFIES (yellow) for the 50k subset and the full MOSES set. GRU models are shown with a solid fill, LSTM models have a dotted fill. Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

Changing Latent Size While Keeping the Hidden Size Fixed

Choosing the latent size of an encoding is always a tradeoff decision. While larger latent sizes increase the encoding's capacity to carry important information, it also inflates model sizes as well as required memory. Inversely, smaller sizes are more memory friendly, but inevitably carry less information. Additionally, choosing a too large encoding size might not help either, since the information content may already be saturated and additional size only holds

redundant or no information. Thus, in this experiment, the goal was to investigate if shrinking the latent size further leads to significant information losses, implying that the baseline's size is already favourable, or if the content could be further condensed without loss of information. I analysed the models' performances when reducing the latent size to 32 as well as to 16 (Figure 23).

On the 50k subset the difference is barely noticeable. As the full reconstruction rate was already around zero, no change was expected in that regard. With respect to the mean similarity, reducing the latent size slightly reduced performance, however only by a few percent points. This effect was particularly mild for LSTMs, where the baseline model achieves mean similarities of 74.8% (SELFIES) and 75.4% (SMILES) and the smallest tested latent size of 16 reduced this only to 74.4% and 73.9% for SELFIES and SMILES, respectively, a reduction of less than 2% in both cases. In GRUs, the reduction was slightly stronger with approximately 4-5%. When training on the full dataset, performance also barely changed for the GRU-architecture trained on SMILES. Here, the smallest latent size of 16 yielded 96.0% mean similarity, while the baseline model achieved 97.6%. For SELFIES, the effect was stronger, with an approximately 9% drop from the baseline to latent size of 16. For the LSTM architecture, performance gradually decreased with smaller latent sizes for both input formats. When looking at the full reconstruction rate, the impact of reducing the latent size on the SELFIES-GRU is severe. It drops from 48.1% in the baseline to ~0% for both sizes, 32 and 16. While the GRU's full reconstruction in SMILES also decreases, the effect is much less extreme. Since the LSTMs full reconstruction rate was very low in the baseline, the reduction by decreasing the latent sizes is observable but small.

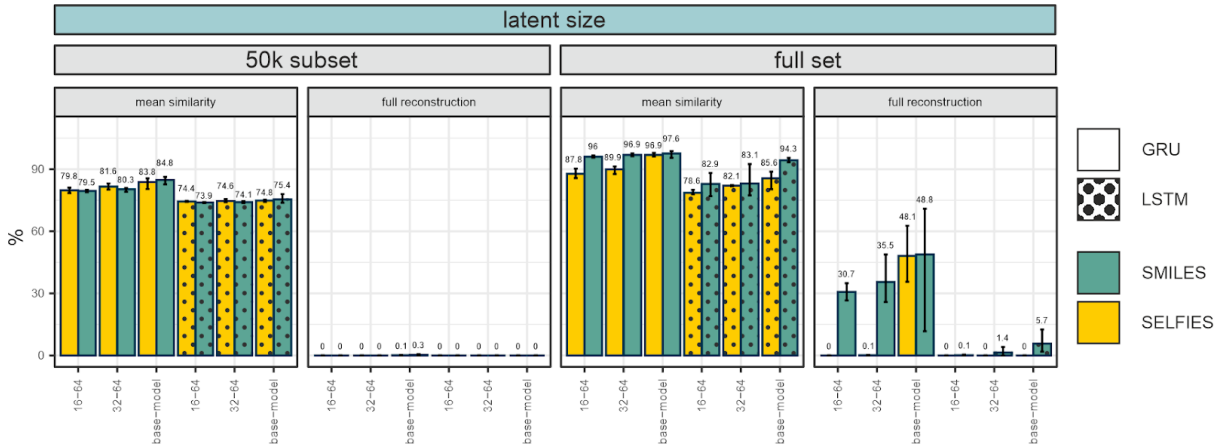


Figure 23: Reconstruction performance of models with manipulated latent size. Results are shown in comparison to the baseline models. Shown are the mean similarity and the full reconstruction of SMILES (green) and SELFIES (yellow) for

the 50k subset and the full MOSES set. GRU models are shown with a solid fill, LSTM models have a dotted fill. Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

As initially mentioned, reducing the latent size bears the risk that the model is not able to store all necessary information, however, increasing it, may inflate model size while not helping model performance due to information saturation in the latent space. To evaluate the latent space utilisation in more detail, a set of test molecules was encoded by each of the models. For the models trained on subsets, these comprised the entire 12,500 test molecules. For the full set, where more than 170,000 test molecules were available, a subset of 12,500 was randomly sampled to make the results more comparable. The encodings were then scaled to a range of $[0,\ 1]$ using min-max scaling, to make patterns comparable across models when visualised as a heatmap.

Different latent space patterns can be manifested by different models, depending on their ability to efficiently store information in the encodings. First, a saturation in latent space utilisation, which occurs when the encoding size offers more room to store information than necessary, would represent itself as a blend of unicoloured stripes and highly variable regions. The unicoloured stripes are dimensions of the encoding that do not carry any information and collapse on a value, because their content does not impact the outcome and they are thus not subject to error correction. Dollar *et al.* [78] refer to this latent space structure as *selective* because only the number of dimensions that are required to represent the data are selected to carry information while excessive dimensions are ignored (unicoloured stripes). Second, a latent space that is not saturated can present itself in two forms: i) if the latent space carries meaningful information for the reconstruction of the input, then all dimensions can be expected to exhibit high variance in their values across the different encodings in order to maximise the number of possible value combinations and thus optimise the utilisation of the available memory. This type will be referred to as the *high utilisation* type; ii) if the latent space does not carry meaningful information for reconstruction, Dollar *et al.* describe a latent space which pattern they termed *posterior collapse*, where the encodings of the different molecules look highly similar, thus, the variance in values held by the different latent dimensions is low and the heatmap exhibits an extensively striped pattern. Examples for the different latent space types are illustrated in Figure 24.

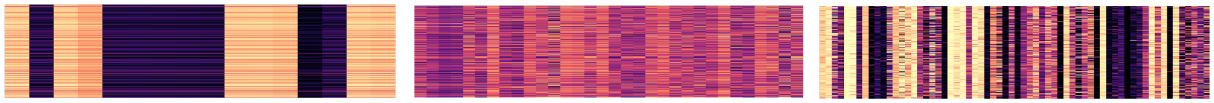


Figure 24: Types of latent space utilisation. The three different types of latent space utilisation patterns found in the models. **Left**: *posterior collapse*, low variance in the latent dimensions, encodings show high resemblance, little information content for reconstruction; **middle**: *high utilisation*, high variance in the latent dimensions, encodings are very different and the information content is high; **right**: *selective*, high variance in most but very low variance in some latent dimension, shows saturation of carried information content.

Indeed, a tendency to all three of these possible patterns can be observed in the models analysed here (Figure 25). For all GRU models, both across SMILES and SELFIES as well as for the subset and the full set, the base model exhibits a *selective* latent space structure with few unicoloured dimensions, indicating that the latent space size of 64 allows high utilisation but also exhibits first saturation effects, which suggests that the ideal latent size lies slightly below 64. This is an expected result, given that the SMILES and SELFIES used for training all have lengths below 64, i.e. no dimensionality reduction that enforces information condensation had been performed. GRU models with lower latent dimensions exhibit the *high utilisation* latent space structure. This aligns well with the observations made with respect to the GRUs performance in mean similarity: Despite reducing the latent size to a half and to a quarter of the baseline, the drop in achieved mean similarity is not equally severe, which is likely due to the efficient utilisation of the available latent dimensions. However, the increased utilisation with shrinking latent size does not fully compensate for the information loss induced by the compression which is clearly illustrated in the earlier described drops in full reconstruction rate.

Interestingly, the LSTMs trained on the subset do not exhibit the *high utilisation* type in any of the experiments. Their baseline models show a *selective* organisation, like the GRUs, but for reduced latent sizes they demonstrate a tendency towards *posterior collapse*, both for SMILES and for SELFIES. An emphasis must be put here on the word *tendency*, because while a strong striping pattern is clearly visible, variances between encodings are still noticeable. This latent space phenotype matches the observations made with respect to performance: the LSTMs are capable of reconstruction to a certain extent, which is represented by their mean similarity rates, however, they underperform in comparison to GRUs, which do not show the tendency towards *posterior collapse*.

LSTMs trained on the full MOSES set are the only ones out of the models compared here, that do not have a *selective* memory type in their baseline architecture, but instead demonstrate *high utilisation*. For lower latent sizes, characteristics of *posterior collapse* as well as *high utilisation* memory types can be observed, with rather strong differences between the different seeds. The fact that the baseline model does not exhibit a *selective*

memory structure indicates that increasing the latent size for these models may offer further room for improvement. Indeed, when increasing the latent size further to 128 the selective structure becomes apparent (suppl. figure 5).

Across all models it can be observed that training on the full set increases the information content in the latent space, which becomes clear by the reduction in *posterior collapse* characteristics, i.e. "stripiness".

Combining the insights gathered from the reconstruction performance analysis as well as the latent space utilisation, the GRUs are much more successful at using the latent space for information perseverance than LSTMs. Not only do they demonstrate high performance in reconstruction in the baseline, but they are also capable of shifting to a *high utilisation* memory mode for smaller latents in order to compensate for the loss in dimension, even when reducing the latent size to 16, which is less than half of the median SMILES and SELFIES lengths. Additionally, GRUs exhibit signs of information saturation in the base models' latent size of 64, whereas LSTMs require larger sizes to accommodate the same information levels, explaining their generally worse performance. It is unclear, however, *why* LSTMs perform more poorly.

Figure 25: Latent space utilisation as heatmaps when modifying the size of the latent vector. The top two rows show results when models were trained on the subset, the bottom two rows show results for the full MOSES set. The three leftmost columns represent models trained on SMILES, the three rightmost columns are models trained on SELFIES. Rows that represent GRU and LSTM models are marked as such on the left. Latent space sizes are stated on top. For each model, three latent space utilisation plots are shown, representing the three different seeds used for training. In each heatmap, rows are molecules and columns are latent dimensions. All encoding values were scaled using min-max scaling within each heatmap to make patterns comparable.

Note, that while GRUs exhibit saturation for the latent size of 64, there is still room for improvement especially when it comes to the full reconstruction rate. Thus, when freezing all other parameters of the base model (hidden size, number of layers, use of attention), a latent size of 64 may exhibit saturation characteristics, but that does not imply that 64 is overall the best possible latent size. As became evident in the previous section, a latent and hidden size of 128 exhibited superior performance. However, this experiment clearly shows that latent sizes smaller than 64 come with a tradeoff in information content, but that GRUs are more capable of compensating for this than LSTMs are.

Increasing the Number of Layers

Increasing the number of layers in neural networks often has beneficial effects when learning different levels of structure in the data. While the first layer can focus on low-level features, other layers can pay increasing attention to more complex features. To investigate if adding more layers is also beneficial when modelling molecules with RNNs, I increased the number of layers from 1 in the baseline model to 2 and 3 layers.

When training the models on the full set, those models that used SMILES as their input format generally increased in performance with an increasing number of layers (Figure 26). While this effect was not so prominent for the mean similarity due to their already high values, the full reconstruction rate substantially increased. GRU performance with only one layer yielded a full reconstruction rate of 48.8%, whereas 2 layers boosted this to 93.2%, adding another layer did not have a significant effect. LSTM performance on SMILES input showed a similar trend as GRU, however, full reconstruction rate with 3 layers showed a strong increase in comparison to 1 and 2 layer(s). Interestingly, the models trained on SELFIES (full set) did not benefit from a third layer. Mean similarity either did not improve or even decreased when adding a third layer to GRU and LSTM. The full reconstruction rate of SELFIES-trained GRUs increased by about 30% from 1 layer to 2 layers, however the third layer reduced this performance almost back to the baseline. Full reconstruction of SELFIES with LSTM remained at 0 for all layer settings.

In the 50k subset, GRU mean similarity and full reconstruction generally increased with increasing number of layers, both for SMILES as well as SELFIES. For the LSTM architecture, mean similarity for SELFIES input increased with the number of layers, while adding a third layer was not beneficial in case of SMILES. Full reconstruction remained 0 for LSTMs in all scenarios.

Altogether, GRUs tend to benefit from 3 layers with the only exception being the full reconstruction rate on SELFIES when trained on the full set. LSTMs on the other hand either show no added benefit when going from 2 layers to 3 layers or even decrease their performance. With the only exception being the full reconstruction rate on SMILES when trained on the full set. Adding layers could not rescue the poor full reconstruction performance of LSTMs on the subset and also not for SELFIES on the full set.

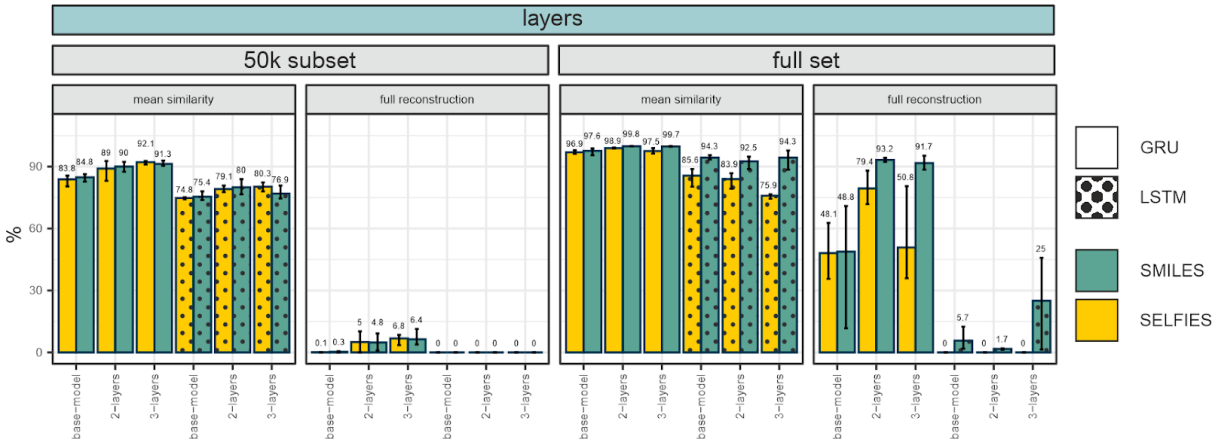


Figure 26: Reconstruction performance of models with manipulated number of layers. Reconstruction performance of models with 2 or 3 layers in comparison to the baseline models (one layer). Shown are the mean similarity and the full reconstruction of SMILES (green) and SELFIES (yellow) for the 50k subset and the full MOSES set. GRU models are shown with a solid fill, LSTM models have a dotted fill. Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

Adding Attention

As previously described, RNNs struggle with increasingly long sequences and attention can often be used to mitigate this issue. Thus, in this experiment, I investigated whether adding attention would also benefit the models when encoding and reconstructing the MOSES set. When training on the full set, mean similarity slightly improved for both LSTMs and GRUs with the exception of GRUs trained on SELFIES (Figure 27). However the improvements were small. More prominent were the changes in full reconstruction rate. Here both architectures strongly benefited from attention when trained on SMILES. Training on SELFIES, though, was either not improved (LSTM) or deteriorated (GRUs). For the 50k subset, slight improvements in mean similarity could be achieved in almost all scenarios (the exception being LSTMs on SMILES), but much like when training on the full set, these improvements were small. With regard to full reconstruction on the subset, GRUs could double their performance on both SMILES and SELFIES, but given their small starting

values the total effect was also minimal. LSTMs showed no improvement in full reconstruction when adding attention.

In summary, adding attention seems to generally have favourable effects, particularly in the full reconstruction rate with the exception of GRUs trained on SELFIES. However, benefits are small for the subset. Dollar *et al.* show in their paper that they observed attention to have a beneficial effect for sequences longer than approximately 60 characters. Given that the molecules used here are shorter, that could explain why the positive impact of the attention is mostly small.

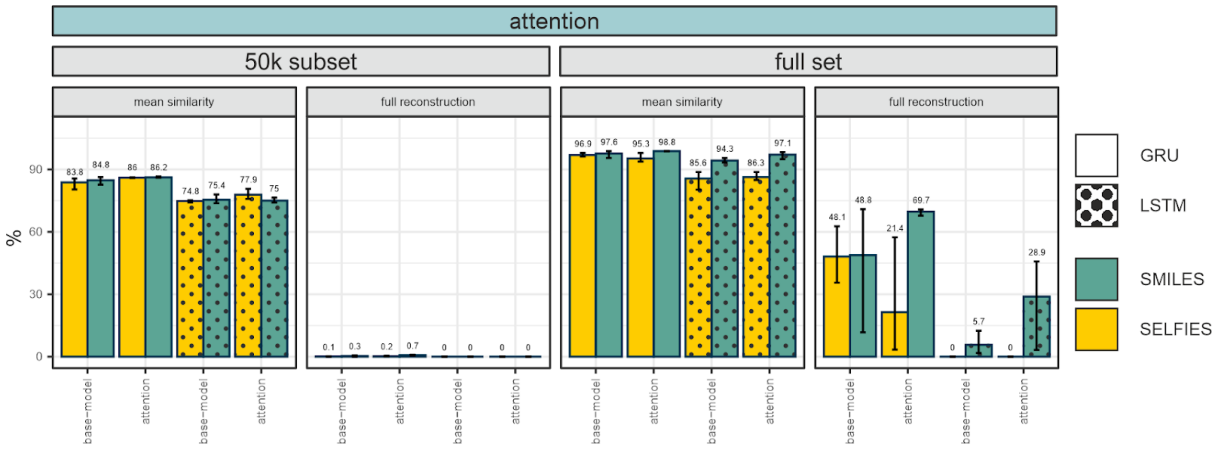


Figure 27: Reconstruction performance of models with and without attention. Reconstruction performance of models with attention in comparison to the baseline models (without attention). Shown are the mean similarity and the full reconstruction of SMILES (green) and SELFIES (yellow) for the 50k subset and the full MOSES set. GRU models are shown with a solid fill, LSTM models have a dotted fill. Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

Improving Subset Performance With Increased Training Time

The results of the experiments described above clearly demonstrate an expected inferiority of the models trained on the 50k subset compared to those trained on the full MOSES set. However, this inferiority was mostly rooted in the full reconstruction rate, while the mean similarity of the subset models already reached high results on some of the architectures. I therefore explored if training the models on the subset for more epochs could close the gap to the models trained on the full MOSES set (Figure 28). I selected the three best performing architectures from the experiments above and trained them for 1,000 epochs. These best performing models across molecule representations based on the two evaluation metrics were: 1) the GRU with latent and hidden size of 128, 1 layer and no attention, 2) the GRU

with latent and hidden size of 64, 3 layers and no attention, and 3) the GRU with latent and hidden size of 64, 2 layers and no attention.

Training these models on the 50k subset for 1,000 epochs indeed improved their performance immensely. Mean similarity for all three models reached values higher than 98% and the full reconstruction exceeded 80% in all cases but one (the third best model on SELFIES reached "only" an average of 73.6%). While the models trained on the full set remained slightly better, increasing the training time on the subset almost fully allowed the models to close that gap. It is worth noting that the subset comprises 97% less data than the full set. This demonstrates that comparable results can be achieved when drastically downsampling the data, making the training much more handleable. Given that the original motivation for downsampling - besides requiring less specialised hardware - was to reduce compute time and make training the models more environmentally friendly, the remaining question now was if increasing the number of training epochs made the training just as time intensive as training on the full set. Timing the training runs yielded that the average training time of the models trained on the full set was 12 hours while the subset required only 8.5 hours on the same machine, a reduction in compute time of almost 30%.

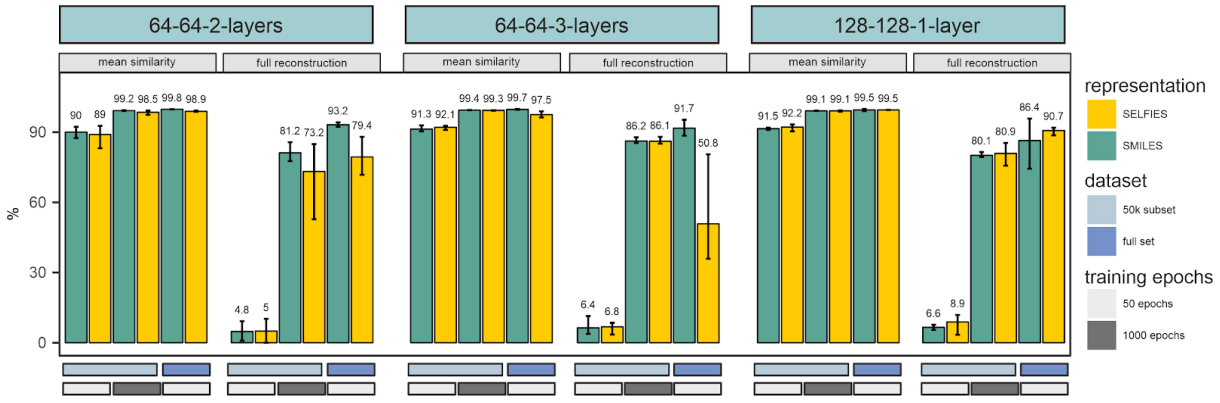


Figure 28: Boosting subset performance with training time. Change in performance when increasing the number of training epochs. Results are shown for the three models that performed best in the previous experiments. For each model, performance in mean similarity and full reconstruction is illustrated for the 50k subset (light blue underline) trained for 50 epochs (light grey underline) or 1000 epochs (dark grey underline) and contrasted with the model's performance when training it on the full MOSES set (blue underline) trained for 50 epochs (light grey underline). Error bars give the performance differences across the three different seeds, mean values are stated above the error bars.

Improving Subset Performance With Additive Optimisation

After investigating the singular effects of isolated architectural changes in the reconstruction performances of the models, the forthcoming question was if the most favourable architectures of the single experiments had an additive effect. Thus, for SMILES and

SELFIES separately, I selected the best performing choices from each experiment and additively combined them into one model each. For SMILES, GRUs generally outperformed LSTMs, 128 gave the best results in the selection of hidden and latent sizes and 3 layers as well as attention also had beneficial effects. Thus the best imputed model for SMILES was predicted to be: GRU, 128 hidden size, 128 latent size, 3 layers, attention. For SELFIES I observed that 3 layers were sometimes detrimental or did not bring added benefit, thus I decided on an optimal inferred architecture of GRU, 128 hidden size, 128 latent size, 2 layers and attention for SELFIES.

The inferred models were trained for 50 epochs for all three seeds. For the best performing seed, training was continued until 200 epochs. Their performance was then compared to the best performing model of the single experiments, which served as a point of reference: The GRU with latent and hidden size of 128, 1 layer and no attention (Figure 29).

Both inferred best models outperformed the reference model The improvement was particularly prominent in the full reconstruction rate. Here, the performance on SELFIES improved from a full reconstruction of 8.9% to 57.0%. For SMILES the improvement was less pronounced, increasing from 6.6% to 36.2%. Training the best performing seeds for 200 epochs pushed the performance even further: The mean similarity reached near perfect scores, 99.2% and 99.3% for SELFIES and SMILES, respectively, and the full reconstruction increased beyond 82% for both models, coming close to the performance of models trained on the full MOSES set. In summary, this demonstrates that the observations made in the separate experiments could be combined to gain additive value and considerably improve the performance of the models.

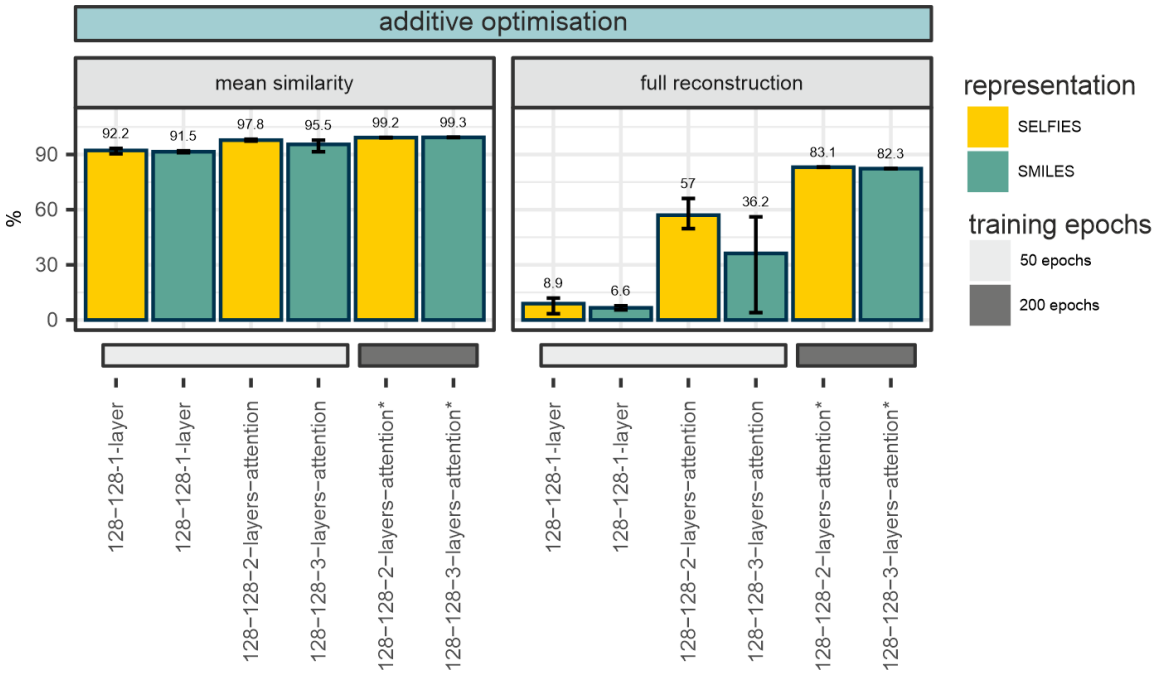


Figure 29: Boosting subset performance with additive optimisation. Results of the additive optimisation experiment. For SMILES and SELFIES, the best performing architectural choices identified in the single-architecture change experiments were composed into an inferred-best performing model. Performance of these two models in both, achieved mean similarity and full reconstruction, after training for 50 epochs (light grey underline) is compared to the performance of the best performing model from the single-architecture change experiments (128-128-1-layer), also trained for 50 epochs. Additionally, training of the best performing seed of the inferred-best models was continued until 200 epochs (dark grey underline).

Latent Space Evaluation

After architectural optimisation as well as increasing the training time to 200 epochs, the devised models show high reconstruction performance in both metrics. However, they are not variational AEs and therefore do not impose any explicit constraints onto the structure of the latent space. I thus determined the chemical quality of their latent spaces and compared it to that of variational autoencoders. Concisely, I compared six different models:

1. The inferred best model for SMILES (GRU, 128 hidden and latent size, 3 layers, attention), trained on canonical SMILES only - **SMILES-AE-can2can**
2. The inferred best model for SMILES (GRU, 128 hidden and latent size, 3 layers, attention), trained using 5-times enumerated SMILES as input and canonical SMILES as the reconstruction targets - **SMILES-AE-enum2can**
3. The inferred best model for SMILES (GRU, 128 hidden and latent size, 3 layers, attention), trained using canonical SMILES as input and 5-times enumerated SMILES as the reconstruction targets - **SMILES-AE-can2enum**

4. The inferred best model for SMILES (GRU, 128 hidden and latent size, 3 layers, attention), trained as a VAE on canonical SMILES only - **SMILES-VAE-can2can**
5. The inferred best model for SELFIES (GRU, 128 hidden and latent size, 2 layers, attention) - **SELFIES-AE**
6. The inferred best model for SELFIES (GRU, 128 hidden and latent size, 2 layers, attention), trained as a VAE - **SELFIES-VAE**

For the second and third model I used enumeration of the input and the target SMILES, respectively. Bjerrum and Sattarov [204] show in their work that enumeration can facilitate latent space organisation in the absence of explicit latent space constraints as they are present in variational autoencoders. All models were trained on the 50k subset for 200 epochs. As previously described in the methods section in more detail, I then enumerated 3 test molecules for the SMILES models and visualised their embeddings using a PCA. For the SELFIES models, I performed random mutations of 3 test molecules and visualised those. During mutation, depending on the chosen token as well as the token it is substituted with, this can in some situations change the molecules drastically. However, in general, the resulting change is small in comparison to the difference between random molecules and this is particularly the case when selecting only the top most similar mutated SELFIES as it was done here, in which case the similarities between the mutated and the original molecules are very high (suppl. figure 6). As can be observed in Figure 30 A, the latent space of the SMILES-AE-can2can model is poorly structured. Enumerations (crosses) and canonical SMILES (dots) of the same molecule do not cluster together but instead are spread across the two components. The latent space in general looks spotted with empty regions, indicating lack of coherency often observed in non-variational autoencoders and something that can be typically mitigated by using VAEs.

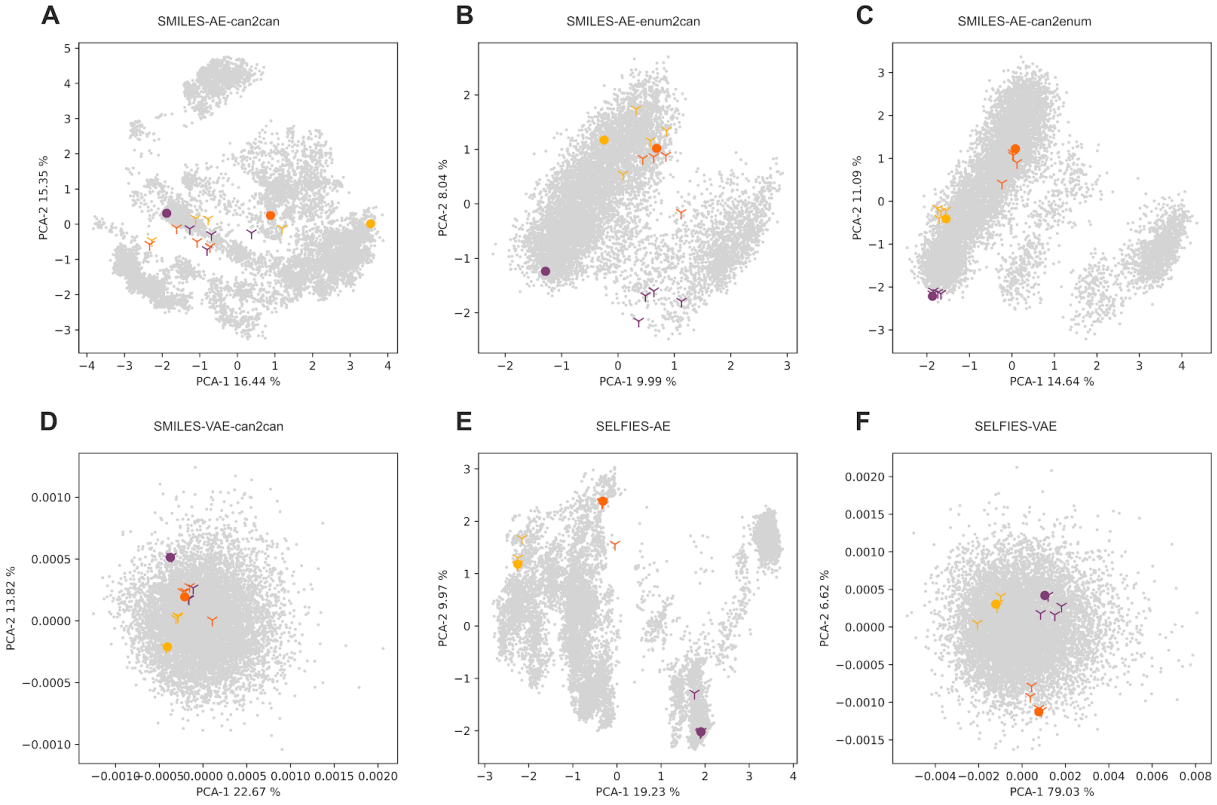


Figure 30: Latent space evaluation for chemical similarity. Each PCA shows the embedding of a molecule (dots) mapped onto the first two principal components. Grey dots represent embeddings of 10,000 randomly selected training molecules. Coloured dots are three randomly chosen molecules from the test set. Respectively coloured crosses are enumerations (in case of SMILES) or point mutated versions (in case of SELFIES) of the original test molecule. Each panel depicts a different model type: **A**) SMILES autoencoder trained on canonical SMILES as inputs and canonical SMILES as reconstruction targets (can2can); **B**) SMILES autoencoder trained on enumerated SMILES as inputs and canonical SMILES as reconstruction targets (enum2can); **C**) SMILES autoencoder trained on canonical SMILES as inputs and enumerated SMILES as reconstruction targets (can2enum); **D**) SMILES variational autoencoder trained on canonical SMILES as inputs and canonical SMILES as reconstruction targets; **E**) SELFIES autoencoder, **F**) SELFIES variational autoencoder.

Training the model with enumerated SMILES as input and canonical SMILES as targets, markedly changes the overall structure of the latent space (Figure 30 B). Besides the fact that it is split into two parts it otherwise appears more homogenous than that of the SMILES-AE-can2can model. Nonetheless, the embedding remains poor, with the enumerations of the molecules being spread far apart. This changes when training the model on canonical SMILES and using the enumerations as reconstruction targets (Figure 30 C). Here, the overall latent space looks similar to that of the SMILES-AE-enum2can model, however the enumeration of the test molecules now cluster very closely to their canonical versions. When comparing the SMILES AEs to their VAE counterpart, the VAE's latent space is expectedly more homogeneous (Figure 30 D). While the enumerations partially overlap, they do not cluster together with their canonical reference in many cases. When training with

SELFIES, enumeration is not an option. Hence, the best inferred model for SELFIES was only trained once as an AE and once as a VAE. Strikingly, while the SELFIES-AE's latent space is also somewhat irregular, the mutated versions of the reference molecules cluster very closely together (Figure 30 E). The fact that these mutations are not different versions of the same molecules (unlike with enumerations), but instead similar, yet distinct molecules makes this even more remarkable. Lastly, looking at the SELFIES-VAE, the latent space again becomes smoother (Figure 30 F), as expected, but in comparison to the SMILES-VAE, the molecules distinctly cluster together in their respective groups. The observations made in the PCAs are further backed up when comparing Euclidean distances within a group versus the distances between random molecules (Figure 31). While for all models the average distance of the enumerations/mutations of the test molecules to each other is smaller than the average distance between random molecules, this distinction is particularly strong in the SMILES-AE-can2enum model and the SELFIES-AE.

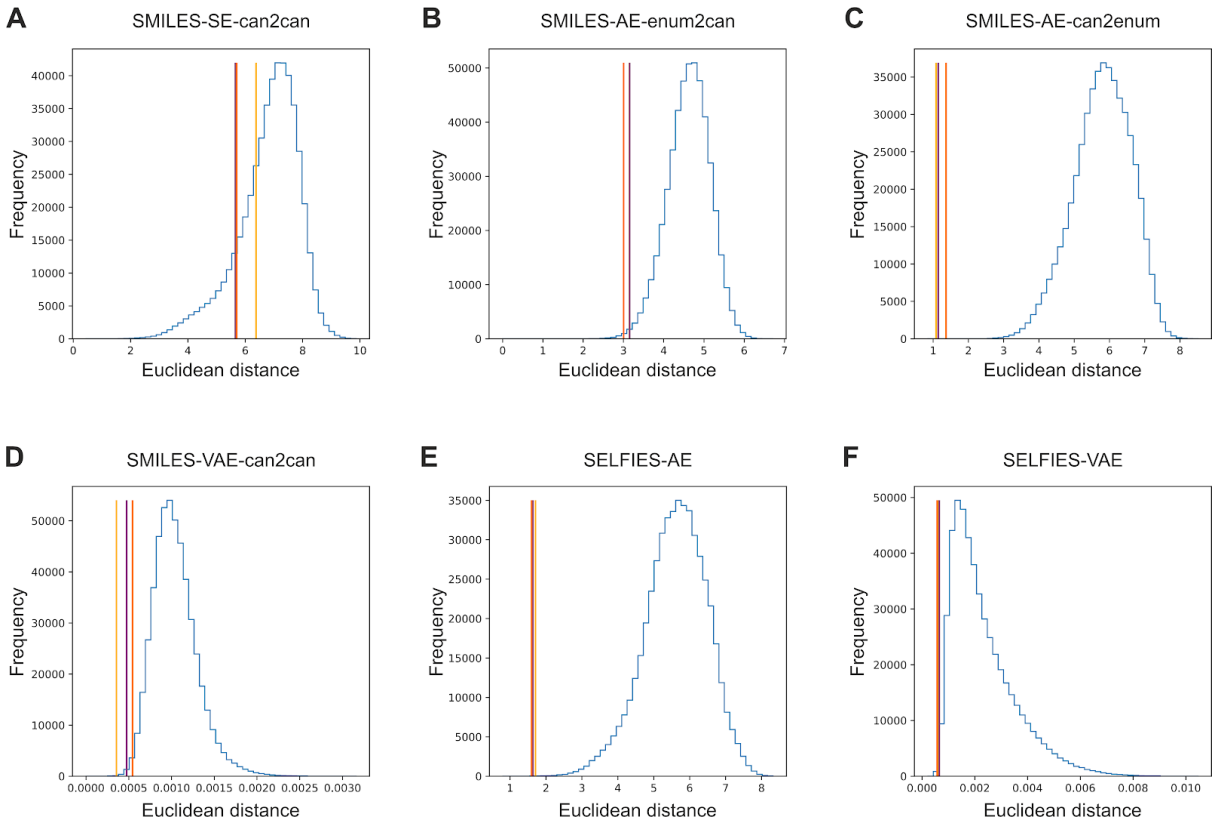


Figure 31: Euclidean distances between encodings of similar versus random molecules. Histograms (blue) show Euclidean distances between encodings of 1,000 randomly selected molecules. Coloured lines indicate mean Euclidean distance between encodings of enumerated (in case of SMILES) or point-mutated versions (in case of SELFIES) of three randomly selected test molecules. Each panel depicts a different model type: **A**) SMILES autoencoder trained on canonical SMILES as inputs and canonical SMILES as reconstruction targets (can2can); **B**) SMILES autoencoder trained on enumerated SMILES as inputs and canonical SMILES as reconstruction targets (enum2can); **C**) SMILES autoencoder

trained on canonical SMILES as inputs and enumerated SMILES as reconstruction targets (can2enum); **D**) SMILES variational autoencoder trained on canonical SMILES as inputs and canonical SMILES as reconstruction targets; **E**) SELFIES autoencoder, **F**) SELFIES variational autoencoder.

In summary, while the SELFIES underperformed for most model architectures in comparison to SMILES with respect to reconstruction accuracy, the chemical understanding that the models gather from them is high enough that their latent spaces, despite its heterogeneity, reflects molecular similarity better than that of a SMILES VAE. In the end, whether an autoencoder or a variational autoencoder is the right choice is strongly task dependent. If the goal is to simply be able to encode and correctly reconstruct molecules, then an autoencoder is sufficient. If the latent space is to be used for downstream tasks that require it to reflect chemical similarities, then SELFIES-based autoencoders have proven themselves as satisfactory. VAEs on the other hand are less suited for tasks that demand encoding and correct decoding given their inherently stochastic nature. The VAEs shown here reach a mean similarity of 77.3% for SMILES and 79.3% for SELFIES, as well as a 0% full reconstruction for SMILES and 0% for SELFIES. Note that the KL-term and the reconstruction term of the VAE loss were weighted with 0.3 and 0.7, respectively, in favour of the reconstruction accuracy. Nonetheless, if the task at hand involves sampling new molecules rather than embedding known ones, VAEs are the only choice out of the two.

3.5 Discussion

The goal of this project was to provide a systematic assessment of the impact that different architectural choices as well as molecule representations have on the reconstruction performance and latent space quality of (variational) autoencoders for molecule embedding. The main motivational factor was to better understand how we can use careful engineering of models to optimise the training process such that the resources required are minimal, without a significant trade-off in embedding and reconstruction quality.

In the experiments, I was able to show that GRUs generally outperform LSTMs on the task of embedding and reconstructing string-representations of molecules. Chung *et al.* [74] empirically compared the performance of LSTMs and GRUs on several datasets of audio sequences and found that there was no difference in the performance of the two model types. These findings, however, are not in disagreement with each other. The differing results can have various reasons. Firstly, Chung *et al.* constructed their models in a way that despite the

varying number of gates, the LSTM models had approximately the same number of parameters as the GRU models. Such modifications have not been made in our experiments, rendering the LSTM models with more learnable parameters than the GRUs. Large parameter sizes and small training sets are known to cause overfitting problems, leading the model to generalise poorly [211]. Secondly, the sequences used by Chung *et al.* are much longer than the SMILES and SELFIES strings used here, thus illustrating the behaviour of LSTMs and GRUs for very different length categories.

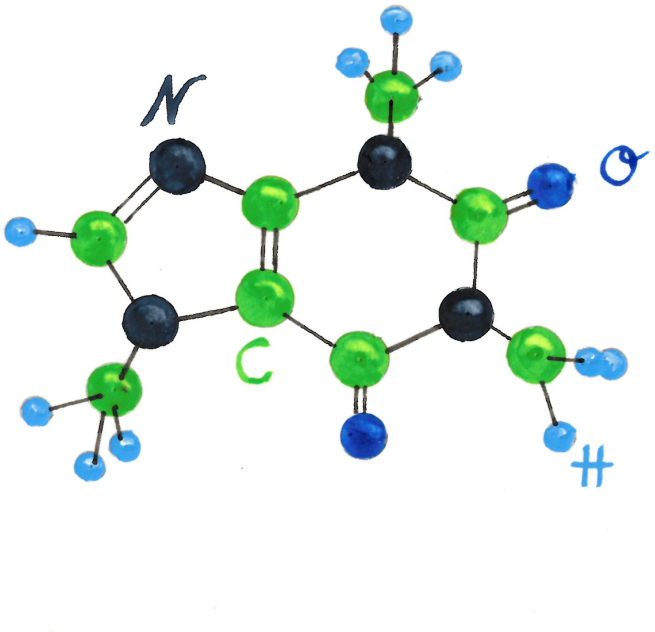
I also found that despite their numerous advantages, SELFIES appear to be harder to correctly reconstruct than SMILES. Initially, this was somewhat surprising given their growing acclaim for deep learning approaches. However, upon further consideration, this finding is not actually unexpected: The fact that SELFIES are always valid, even when their tokens are randomly assembled, entails that there are no syntactic rules for the model to learn from. In SMILES, there are some token sequences that the model will not encounter because they do not represent valid chemistry. In SELFIES on the other hand, this is not the case. It is not difficult to imagine how this makes the learning task much harder. In addition to that, SELFIES tokens do not always have the same meaning, which is another distinct difference to SMILES. Instead, SELFIES tokens represent a rule which dictates the next atom, bond, branch or ring, depending on what came before. This ambiguity paired with the complete absence of rules of invalidity could explain why the models have a harder time learning how to reconstruct them. At the same time, it also offers an explanation on why, despite these reconstruction problems, the latent spaces of the SELFIES models were more chemically meaningful than those of the SMILES models: The SELFIES tokens inherently encode rules of chemical validity and therefore a broader contextual description of the molecule. This makes SELFIES particularly interesting for generative models. Any generated molecule is guaranteed to be valid, which allows the model to fully focus on understanding the underlying chemistry rather than the syntax rules.

Additionally, the experiments showed that thoughtful model design has a strong impact on performance and resource consumption. By interpolating the findings from the singular architectural manipulations into an additively optimised model architecture in combination with increased training epochs I was able to achieve comparable performance to the full set models while using only 3% of the data and reducing the required training time by almost 30%. The implications of these insights are two-fold: Firstly, the ever increasing energy consumption caused by training machine learning models is a serious threat to our environment. Efforts for green computing and particularly green AI are therefore more

needed than ever. The work presented here provides evidence that models trained on less data and for less time can be competitive. Secondly, the results presented here have implications for the democratisation of AI. In order to make AI ubiquitously available and not only to extremely wealthy companies and institutions, the models must be trainable and accessible on non-specialised hardware. To realise this, training budgets with respect to required memory and consumed energy must be reduced, a task that can be achieved by architectural optimisation.

While the results presented here indicate that low-resource and high-quality training are not mutually exclusive and can be harmonised by using a refined model architecture, more research has to go into how to do this efficiently. The computational overhead of training all the models presented here to engineer an optimised architecture does not align with the overall task of saving resources. Thus, this approach is not a viable option to tackle this kind of task in the future. However, it demonstrates that there exist systematic connections between architectural parameters and solving the machine learning task at hand. Therefore, more research is required towards explaining these connections and translating them to other machine learning tasks and model types. This falls into the field of explainable AI and is a crucial step to saving energy and democratising AI by building a suitable model for a task rather than the presently often encountered approach of compensating generic model design with larger data and longer training.

4. **DrugDiff** - Guided Generation of Molecules with Desired Target Properties



4.1 Introduction

The costs of drug development are ever increasing [212,213], burdening the pharma-companies and reducing the prospects of disease curing treatments. The cost increase is fuelled by expensive development processes and drug trials [153] and a yet decreasing yield of eventually approved commercial products. Thus, new solutions are needed that narrow down the number of proposed drug-leads to a small subset with desired target properties.

Learning molecular structures for property prediction or other discriminatory tasks requires the extraction of those features necessary to make the prediction. Depending on the complexity of the predicted property and its underlying molecular features, this is not always an easy task. However, *generating* molecules is yet much harder. It requires the model to learn a very complex distribution representing all the features necessary to build valid and diverse molecules.

The generative process is further complicated when the generation is conditioned on a target property. Conditional molecule generation is extremely important for real world application of these models. To narrow down the pool of drug leads, i.e. small molecules that have undergone some optimisation towards one or several target properties, and therefore make the drug development process more efficient, guidance in the generation process is essential.

Following the progress in the field of generative modelling, different model types have been applied to the task of *de novo* molecule generation [19,78,173,175,177,178,180,181,205,214,215]. Although the focus lies mostly with physico-chemical properties, there are some notable publications that consider more complex systems-biological properties, such as induced gene-expression: Pham *et al.* [177] use a conditional VAE to generate molecules based on gene-expression profiles, Born *et al.* [178] achieve this specifically in the context of anti-cancer drugs using VAEs and reinforcement learning, Shayakhmetov *et al.* [179] use a Bidirectional Adversarial Autoencoder to generate molecules based on gene-expressions and vice versa and Méndez-Lucio *et al.* [171] use a conditional GAN for gene-expression-based molecule generation. Zhavoronkov *et al.* [180] propose a Variational Autoencoder coupled with reinforcement learning and while they do not consider gene-expression they stear molecule generation towards a designated protein target. Westermayr *et al.* [181] propose an autoregressive model coupled with a property predictor that generates molecules for material design with desired properties by iterative retraining of the model with a biassed subset. However, all of these models come with a

limitation: They are trained in a way that adding additional properties would require retraining of the model which hinders the expansion towards multi-property optimisation. A flexible plug-and-play dynamic that allows addition of further properties without retraining was considered by Eckmann *et al.* [174] and while they included generation of ligands for protein targets as a condition, they did not consider gene-expression.

In this project, I will reuse some elements of the work by Méndez-Lucio *et al.* [171] and Eckmann *et al.* [174] and explicitly compare the performance of our model with theirs. Hence, their work requires a more detailed introduction: Méndez-Lucio *et al.* implemented a conditional GAN, that receives Gaussian noise as input and a gene-expression vector of 978 landmark genes as defined by the L1000 CMap [202] database as a condition. The cGAN then generates a latent representation of a molecule based on the condition. This latent representation can be translated back into a molecule using a pre-trained SMILES-to-grammar autoencoder model. During training of the model, the feedback of a discriminator network that receives the generated latent representations and predicts if it is real or fake, is used to train the generator. Additional training feedback comes from a predictor *f*. This predictor was trained to receive a molecule and a gene-expression vector as input and return a score of how likely the molecule induces that gene-expression profile. The feedback of this network is meant to ensure a meaningful connection between the generated molecule and the condition. For brevity, this model will from now on be referred to as *cGAN*.

Eckmann *et al.* [174] developed a Variational Autoencoder to embed, reconstruct and sample SELFIES strings. To generate molecules with a desired property, they then connected the trained decoder in series with an independently pretrained property predictor. After sampling from the VAE's latent space, the molecules were decoded and passed to the property predictor. The molecules are then optimised towards the target property by inceptionism-like reverse optimisation, i.e. the backpropagation from the output continues all the way to the input - in this case the latent space of the VAE - and manipulates the input to better match the desired output. They used both physico-chemical property predictors as well as predictors for the binding affinity to two protein targets. For brevity, their work will be referred to by their project acronym LIMO from now on.

Despite the impressive progress demonstrated in these works, there is still a lot of room for improvement with respect to multi-conditioning and complex systems-biology based properties as for example compound induced gene expression.

The goal of this project was therefore to build and improve upon the existing work and put a particular emphasis on complex conditioning using systems-biology properties such as

binding to a target property or induced gene-expression and on multi-property conditioning, all while finding a framework that avoids retraining of the model when adding new properties. To this end, I worked together with a consultant from Helmholtz AI and we developed a generative model based on latent diffusion and classifier guidance that successfully generated valid and diverse molecules in a conditional setup.

Conditional generation of data instances using diffusion models has been very successful in other domains, especially text-conditioned image generation. In this context, especially Stable Diffusion [28] and OpenAI's DALL-E 2 have attracted a lot of attention. Stable Diffusion is based on a latent diffusion model conditioned on text embeddings. DALL-E 2 as well is a diffusion model using text embeddings provided by Guided Language to Image Diffusion for Generation and Editing (GLIDE). A demonstration of these models' capabilities can be found below in Figure 32. The success of these models motivated us to use them for our molecule generation task. Diffusion models have been used on 3D-coordinates of molecules [216,217], however the conditioning was part of the training procedure, requiring the model to be retrained for each condition and making the training process entirely dependent on labelled molecule data, they did not allow multi-conditioning and did not include complex, high-dimensional biological properties.

Figure 32: Generating images with diffusion models. Left: An image generated by the Stable Diffusion 2.1 Demo published by Hugging Face when prompted with creating a coal drawing of a female PhD student writing her thesis on a computer. Right: An image generated by OpenAI's DALL-E 2 when prompted for a watercolour drawing of an excited PhD student handing in her thesis.

Note that I originally planned to use a cycle-GAN [218] for this project. Cycle-GANs were developed for tasks where the training goal is a mapping between two domains, but where no - or only very few - paired data between the domains exists. They comprise two GAN models that are interlinked using a forward and backward cycle-consistency constraint that enforces one-to-one mappings and therefore consistent translation and retranslation between the domains. Given the limited availability of publicly available paired gene-expression data and their inducing compounds, but a vast amount of unpaired molecular and gene expression data, this approach was compelling for the problem at hand. However, there were three main reasons why we eventually decided against them. Firstly, although they were designed to allow training on unpaired data of different domains, their original application area of images renders the two domains not entirely distinct: While the artstyle of an image may change or a summer scenery is transformed into a winter landscape, the underlying structure of the image is typically maintained, providing a lot of guidance to make the two versions of an image match. This is not given in our case since the molecule domain and the gene-expression domain are truly different in structure despite there being an underlying relationship between them. To use cycle-GANs in our application scenario, additional components would be required to overcome this hurdle. Secondly, the enforced one-to-one mapping of these models is not in our interest when it comes to generating molecules. For real-life applicability, one would be interested in getting a variety of different candidate molecules that fit a condition, not only a single one. The third reason is that the cycle would allow the generation of molecules based on one other domain. It is, however, not clear how such a framework would be extended to incorporate several molecular properties in the generative process, particularly how to do so without retraining the model for any new constellation of properties. More information on cycle-GANs and the reasons why we decided against using them can be found in the appendix.

4.2 Methods

4.2.1 Model

The generative model in its entirety consists of three parts: 1) a variational autoencoder that embeds the molecules into a latent space; 2) the diffusion model and 3) a set of classifiers that is used to guide the generative process (figure 33 A).

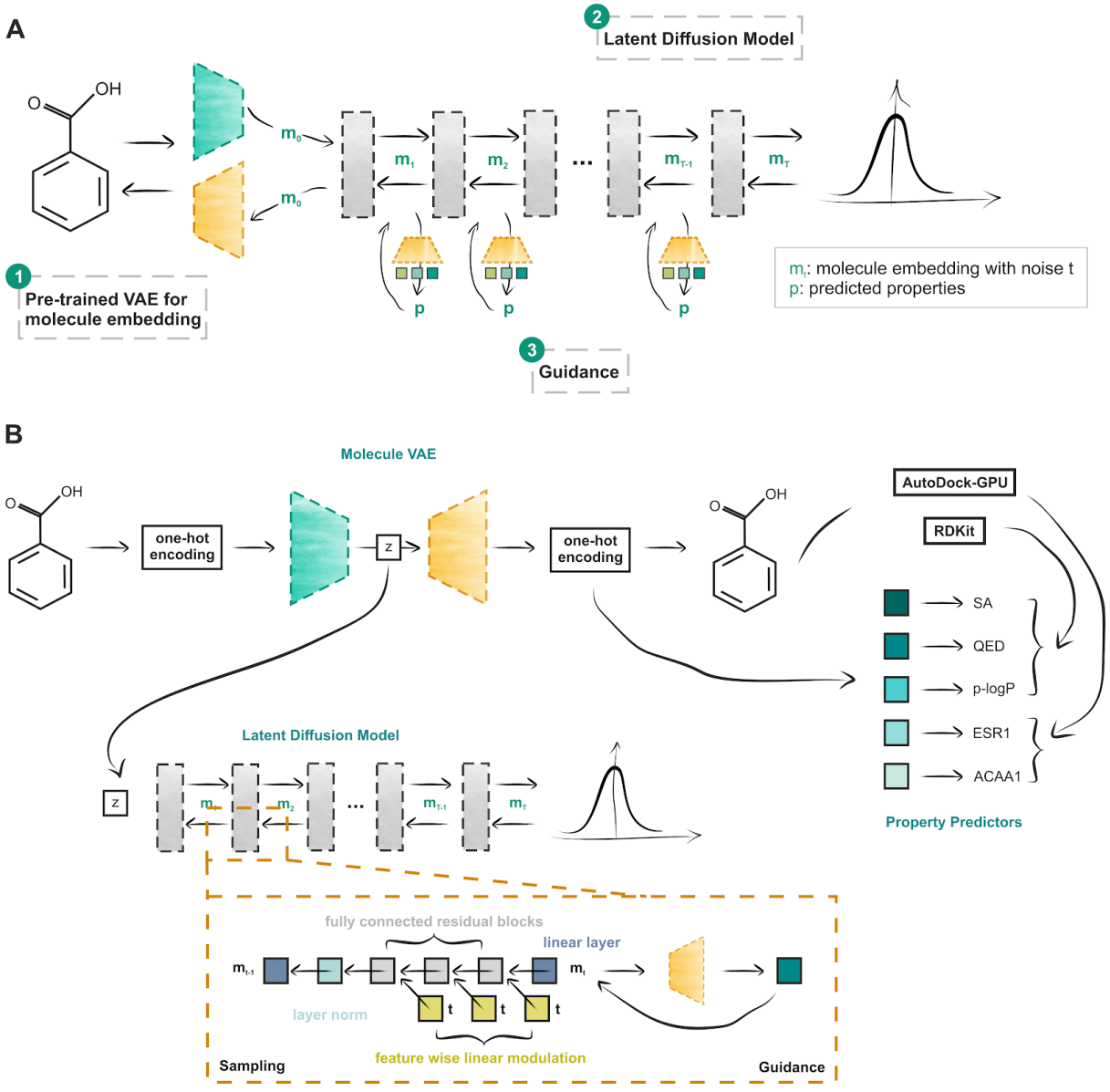


Figure 33: Overview Drug Diffusion Model. **A)** Coarse-grained overview of the three parts comprising the drug diffusion model: 1. The molecule-VAE, 2. The latent diffusion model, 3. The guidance step using property predictors. Molecules are first embedded into a latent representation by a pre-trained molecule-VAE. Then, a latent diffusion model is trained without conditions on the molecule latents. Lastly, during generation, guidance towards desired properties is provided by separately trained property predictors. **B)** Detailed view of the different building blocks. On top is the molecule-VAE, the latent space of which is used to train the diffusion model. The concise composition of the diffusion model is illustrated in the orange box on the bottom. A sketch of the training procedure of the property predictors is illustrated on the right. Guidance using property predictors (square with dark green fill) during the sampling procedure is also illustrated in the orange box.

Variational Autoencoder

As mentioned in the introduction, molecules have to be transformed into a format that makes them processable by a deep learning model. Here, I embedded the molecules into numeric vectors using a VAE as it is described in [174]. The VAE uses one-hot encoded SELFIES as input. The use of SELFIES guarantees that all vectors sampled from the VAE's latent space

decode to a valid molecule. For the experiments presented here, I trained the VAE once on the Zinc250k dataset and once on the ChEMBL-22 dataset. Zinc250k is a subset of the ZINC database [209,219], and contains approximately 250000 small, drug-like molecules. ChEMBL is a database that collects bioactive and drug-like molecules. These datasets were used in the related work by Eckmann *et al.* [174] and Méndez-Lucio *et al.* [171], respectively, and were utilised here to allow direct comparison of model performances. I also adapted the sizes of their molecule embeddings, which were 1024 and 256, respectively and removed all molecules from the ChEMBL-22 dataset that exceeded a length of 120 characters in their SMILES representation. This was according to the filtering procedure described in [171]. The VAEs were each trained for a total of 18 epochs.

The diffusion model was subsequently trained with the embeddings of the molecules present in the respective database.

Classifiers

I used a series of classifiers to guide the diffusion process in order to generate molecules with desired properties. Most of the classifiers were recycled from the *LIMO*-project. They were used to impact the molecule generation with respect to physico-chemical properties, such as synthesisability (*SA*, low is good), drug-likeness (*QED*, high is good) and solubility (*logp*, high is good). As described in [174], these classifiers were not trained on the latent representations of the molecules, but instead on their decoded one-hot-encodings. As pointed out by the authors, this improves the learning process by discretising the otherwise infinitely large latent space of the VAE. For this project, this came with the advantage that I did not have to retrain the predictors in a time step-dependent manner as usually required for classifier guidance [97]: The noisy intermediates sampled from the diffusion model at the different time steps were first passed to the decoder network which is inherently capable of handling noisy inputs sampled from a Gaussian. The decoded samples were then run through the predictors, independent of the time step.

To incorporate biological properties into the generation process, I distilled the classifier described in the *cGAN* paper, which provides a score on how likely a given molecule induced a given gene-expression profile. The predictor was trained on the consensus signatures of so-called ‘Landmark’-genes of the LINCS L1000 dataset [202]. For the generation of this dataset, the expression of approximately 1000 genes - the Landmark genes - was measured after treating a variety of cell lines with different compounds at different concentrations and for varying incubation times. These genes were selected based on the observation that the

expressions of many other genes can be imputed based on their expressions, as described in the original publication [202]. Replicates of the induced expression profiles are averaged into consensus signatures. To train the predictor, the *cGAN* authors filtered the samples for two cell lines (MCF7 and VCAP), an incubation time of 24h and compound concentrations of 5 and 10 μM.

However, the original predictor was incompatible with our model since it was 1) implemented in TensorFlow unlike our model which is implemented in PyTorch, and 2) it was trained to receive latent embeddings of the molecule. I decided against using latents as input, since - as outlined above - training predictors on VAE-latents is a harder task to learn than to train on the decoded one-hot encodings of the molecules [174]. Hence, I used a process referred to as *knowledge distillation* [220] to transfer the predictor to PyTorch and adapt a different input format for the molecule. During knowledge distillation, a model is trained to learn the predictive probability distribution of another model. A nice analogy for this process can be found in this blog [221]: The model that is being distilled is like a teacher, who is very familiar with the concepts explained in an extensive textbook. The textbook in this analogy represents a large training dataset. The model that we are distilling into can be viewed as a student. The student could learn directly from the textbook, but would likely have a hard time, given lack of experience and prior knowledge. However, if the student instead learns from the teacher, they will gather a significant amount of knowledge without having to read and understand the entire book.

More formally, during model distillation, a new model is trained to predict like an already trained model using a KL-divergence loss on its own predictions versus the predictions of the pre-trained model. Normally, this technique is used to distil the knowledge of very large models into smaller ones that are easier to deploy (e.g., on mobile devices), however, here I am using it to utilise the knowledge of the predictor while adapting to the requirements of our model architecture.

To this end, I sampled 100,000 molecules from the VAE trained on ChEMBL-22, decoded them into their one-hot encodings and translated them to SMILES. The SMILES were then embedded using the Autoencoder of the *cGAN*-project. I randomly sampled gene-expression profiles from the L1000 dataset to generate predictions with the original classifier kindly provided by the authors, using the embedded molecules and the sampled profiles. Those predictions served as the target values to train our classifier. To evaluate the similarity of the distilled predictor to the original, the coefficient of determination (R^2) between the

predictions on a hold-out test set not used during training was calculated and the predictions were plotted against each other. The results indicated concordance (suppl. figure 7).

For use with the ChEMBL-22-based model, the *LIMO*-predictors were retrained also by sampling 100000 training molecules from the autoencoder. This was necessary since the one-hot encodings are of dimension (vocabulary-length) x (maximum molecule length). These dimensions vary between then VAE trained on Zinc250k and that trained on ChEMBL-22.

For the *ESR1*-ligand generation taks (experiment 4), the *ESR1*-binding affinity predictor was retrained on the full Zinc250k dataset to improve performance in comparison to the original classifier from LIMO which was trained only on 10,000 molecules. The target affinity values for the training procedure were acquired by running docking simulations with AutoDock-GPU (version 09773678fc7e39677061d765b767f4bae8930fb7-dirty) [222]. Therefore, all SMILES from the Zinc250k data dataset were converted to the PDBQT (Protein Data Bank, Partial Charge, Atom Type) file format using Open Babel (version 3.1.1) [223] as required by AutoDock-GPU. The chosen method for computing partial charges was the Gasteiger method. The pH was simulated as 7.4 and the grid maps for the target protein (*ESR1*) were taken from the LIMO repository (LIMO/1err at main · Rose-STL-Lab/LIMO (github.com)). Docking was performed 5 times for each molecule and the lowest affinity was chosen as the target value for training. The predictor was trained for 50 epochs.

Diffusion Model

The type of diffusion model chosen for this work is a *latent* diffusion model [28]. The underlying model architecture is a denoising diffusion probabilistic model (DDPM) [34]. The network comprises a linear layer followed by three fully connected residual blocks, a layer norm and an additional linear layer (figure 33 B). The current timestep at each pass is embedded by feature-wise linear modulation layers and then passed to the fully connected residual blocks. The input and output size of the network differed for the two datasets used and was set to 1024 for the Zinc250k set and to 256 for the ChEMBL-22 set.

4.2.2 Training Procedure

In every training step, a batch of molecules is embedded using an auxiliary variational autoencoder. For each sample in the batch, a time step t is sampled uniformly from $[1, ..., T]$.

Using the *reparameterization trick* explained above, each sample is noised with Gaussian noise according to the sampled time step by retrieving β_t from the variance schedule. The model then predicts the noise that has been added, given the noised sample and the time step. Mean-squared error (MSE) loss is computed between the true added noise and the predicted added noise. The model is trained to minimise this loss.

4.2.3 Sampling Procedure

Sampling the diffusion model works by traversing it in reversed order: Starting with sampled Gaussian noise at time step T, the model predicts and removes the noise added during the forward process at time step $T - 1$. With each step, the sample becomes less noisy until eventually a sample from the original data distribution is produced.

4.2.4 Optimisation Procedure

Optimising the molecule towards a desired property happens by interfering with the sampling procedure at every time step. The noisy latents are decoded using the auxiliary VAE's decoder and passed to one or several pre-trained property predictors. The predictions are then used to calculate the gradient with which to manipulate the latent sample such that the prediction becomes better. Concisely, that means that the noise which the model removes during the sampling step is modified using the computed gradient, impacting the input to the next time step.

4.2.5 Evaluation Metrics

Novelty refers to the percentage of generated molecules that were not present in the training data set, **uniqueness** is the percentage of generated molecules that were generated only once, **validity** is the percentage of valid molecules and **diversity** was computed using the function *internal_diversity()* from the *molsets* package. As described by the authors [208], this function assesses chemical diversity of a set of molecules. The metric allows detecting mode collapse, a common failure type of generative models. Mode collapse refers to when the model fixates on a small subspace of the learned domain and ignores other areas, resulting in highly similar generated samples.

4.3 Results

Experiment 1: Unconditional Diffusion Model

I first trained two unconditional Diffusion models, one on the Zinc250k and one on the ChEMBL-22 dataset, using the training procedure described above. Each model was trained for 100 epochs and at the epoch with the best performance, the model was saved. To evaluate whether or not the models were able to capture the full spectrum of molecules presented to them during training, I sampled 10,000 molecules from each diffusion model and its corresponding VAE. I then computed the novelty, uniqueness, validity and diversity for the sampled molecules. The results are presented in Table 4.

	Novelty [%]	Uniqueness [%]	Validity [%]	Diversity [%]
DM (Zinc250k)	**98.49**	**98.49**	**100.00**	89.00
LIMO (Zinc250k)	98.20	98.20	**100.00**	**91.00**
DM (ChEMBL-22)	**97.32**	**97.82**	**100.00**	91.00
LIMO (ChEMBL-22)	97.16	97.69	**100.00**	**92.00**

Table 4: General statistics of molecules sampled unconditionally from the diffusion model and LIMO. Novelty, Uniqueness, Validity and Diversity of 10,000 molecules sampled from the unconditional diffusion model (DM) and the LIMO VAE trained on the Zinc250k and the ChEMBL-22 data set.

The differences between the diffusion model and the corresponding VAE are generally minimal. The validity is always 100% which is expected given the SELFIES guarantee to always produce valid molecules. The diffusion model outperforms slightly in novelty and uniqueness, but shows slightly reduced diversity. Altogether, the differences are negligible.

Key chemical properties (SA, QED and logp) were calculated for all sampled molecules as well as the original training data sets. As shown in Figure 34, the diffusion models capture the entirety of the value ranges for the three properties. Nonetheless, the distribution means resemble those of the VAEs more closely than those of the data the VAEs were trained on. This is expectable, since the VAE embeddings were used to train the diffusion models.

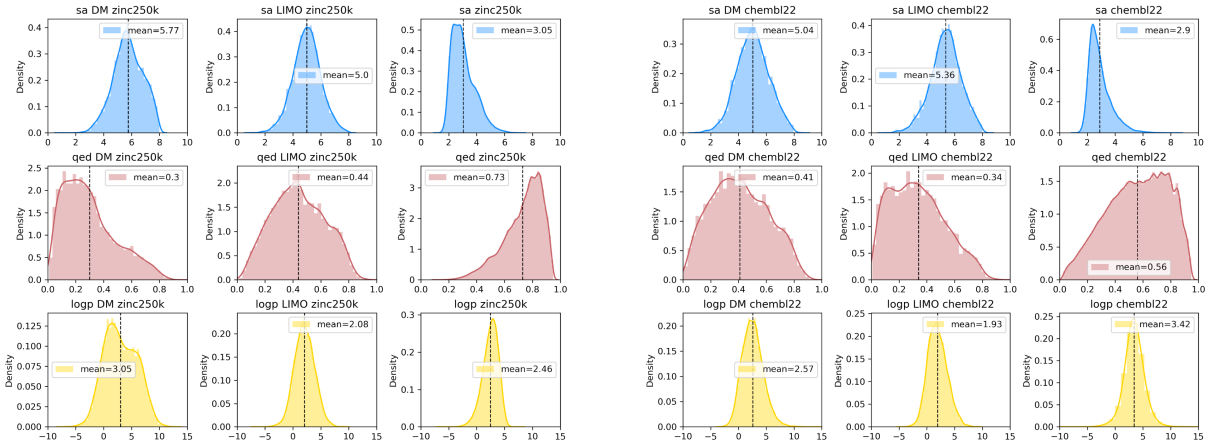


Figure 34: Properties of molecules generated by the unconditional diffusion model. Synthetic Accessibility (sa), drug-likeness (qed) and solubility (logp) of the 10,000 molecules sampled from the unconditional diffusion model (DM; left column) and the LIMO VAE (middle column) as well as all the molecules of the full original data (right column) used for training for both, training on the Zinc250k (left) and the ChEMBL-22 (right) dataset.

Experiment 2: Single-Property Guidance

To assess the susceptibility to guidance in the generative process, I first manipulated the diffusion model with individual properties. I used the property predictors provided by *LIMO* to target the generation with respect to *SA*, *QED* and *p-logp* (*penalised logp*, a metric that promotes high *logp* values but penalises low synthesisability and cycles with more than 6 atoms). I sampled 10,000 molecules with and without guidance for each property. Best results for these three properties were achieved with different guidance scales which I chose using hyperparameter search. Using guidance scale 50, the mean synthesisability could be decreased from 5.77 in the unconditional scenario to 2.95 with guidance (Figure 35 A). Drug likeness (*QED*) could be increased from an average of 0.3 without to 0.55 with guidance of 50 (Figure 35 B) and p-logp from -4.14 to 3.11 with guidance 100 (Figure 35 C). In all cases, the internal diversity remained high: 81.88%, 88.83% and 81.10% for *SA*, *QED* and *p-logp*, respectively.

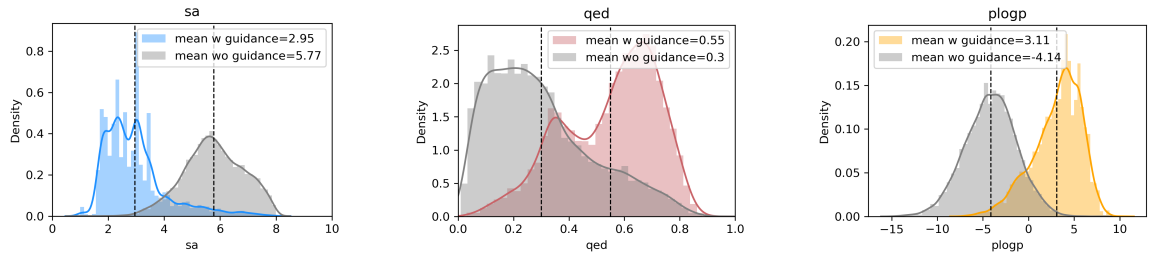


Figure 35: Results of the single-property guidance experiments. Shown are properties of 10,000 molecules sampled without (grey) and with guidance (colour). The properties used for guidance are **left**) synthetic accessibility with guidance

scale 50 (blue), **middle**) drug-likeness with guidance scale 50 (red) and **right**) penalised logP with guidance scale 100 (yellow).

Experiment 3: Multi-Property Guidance

Single-property optimisation only has limited applications in praxis. Much more often multiple properties are to be targeted at once. Thus, in this experiment I addressed two scenarios: 1) given two target proteins, can the model be guided to generate molecules that demonstrate high binding affinity to both, while keeping the SA score low and the QED high; and 2) can this also be achieved for specifically binding one but not the other protein? For the protein choice I took inspiration from LIMO: The authors chose *ESR1* (Human Estrogen Receptor), known for its involvement in breast cancer. The protein is well studied and some inhibitors are known. The second protein they chose is *ACAA1* (Human peroxisomal acetyl-CoA acyl transferase 1), an enzyme. The motivation they gave for choosing this protein is the lack of known inhibitors. A crystal structure is available and the enzyme is suspected to have a drug binding pocket.

For each scenario, I generated 10,000 molecules, both with our model and with the original LIMO model for a point of comparison. I stuck with the importance-weighting regarding the signal of the four properties as used in LIMO: weights 5, 5, 2, -8 for *ESR1*, *ACAA1*, SA and QED, respectively, in the first scenario and for the second scenario 5, -5, 2, -8. Figure 36 shows the molecules sampled for each scenario after filtering for SA < 4 and QED > 0.6 to guarantee good synthesisability and high drug-likeness. For the first scenario - targeting both proteins (Figure 36 A) - the median affinity score is distinctly lower for the molecules generated with our diffusion model than those generated with the LIMO model. A lower score denotes higher predicted affinity to the target. In the second scenario - targeting *ESR1* but not *ACAA1* - the desirable outcome was a low score for *ESR1* and a high score for *ACAA1*. As demonstrated in Figure 36 B, the LIMO model is able to create a noticeable difference in mean predictions, however, our diffusion model achieves this to a much greater extent, creating molecules that are predicted to be much more specific to ESR1.

The molecule generation with LIMO was run with its default number of optimisation steps, which is 10. This means that for every sampled molecule, the latent is iteratively manipulated 10 times using the gradients from the property predictors. Our diffusion model only performs one optimisation - i.e., guidance - step per diffusion step. However, given the overall different architectures and the fact that our diffusion model generates the sample not in one step (as LIMO does), but instead gradually over 1000 steps, I sampled LIMO again with different step

numbers (100, 1000, 10000) to investigate how far the model could be pushed to extreme performance. The results are illustrated in supplementary figure 8. While LIMO comes close to our diffusion model's performance when running 10000 optimisation steps in the first scenario, the diffusion model's performance in the second scenario (targeting one protein, but not the other) still greatly outperforms LIMO with 10000 steps.

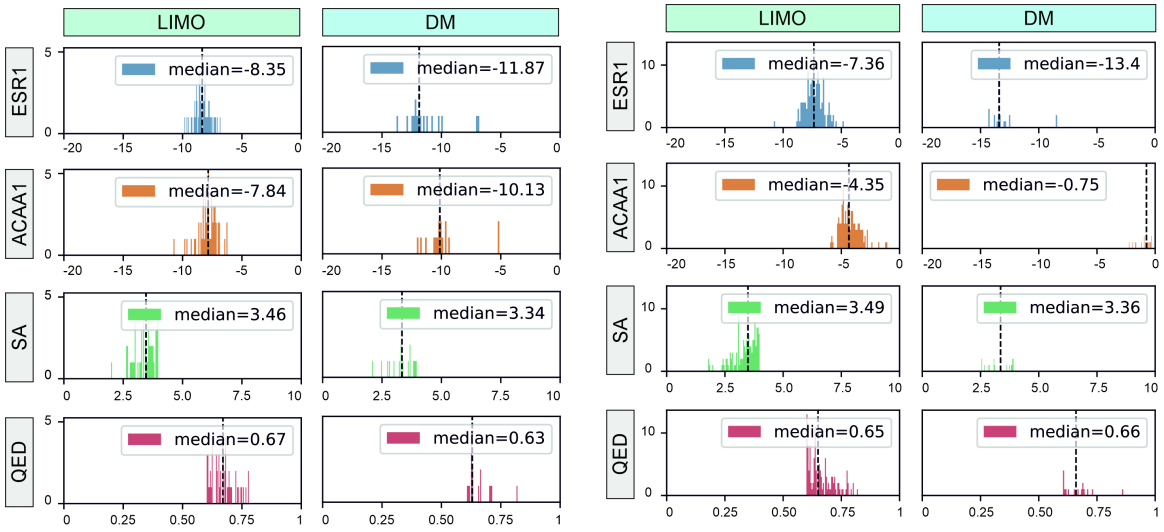


Figure 36: Multi-property guidance. Shown are predicted *ESR1*- and *ACAA1*-scores as well as computed synthetic accessibility (SA) and drug-likeness (QED) of molecules generated with LIMO and with the diffusion model (DM). All generated molecules were filtered for good synthesisability (SA < 4) and high drug-likeness (QED > 0.6). A) Results of scenario 1, where generated molecules were meant to show high affinity (represented by a low score in the plots) to both target proteins; B) results of scenario 2, where generated molecules were meant to have high affinity to *ESR1* (represented by low predicted score) and low affinity to *ACAA1* (represented by high predicted score).

Experiment 4: ESR1 Ligand Generation

The predictors used for guidance in the previous experiment were taken from the LIMO repository. Those predictors were only trained on 10,000 molecules since the generation of the training set, i.e. retrieving the binding affinities for all training molecules to the protein target via docking simulation, is computationally expensive. That was sufficient to generally demonstrate the model's successful behaviour when being guided using multiple complex properties. In this experiment, however, I was interested in applying the diffusion model to generate candidate ligands for the *ESR1* target. Such a task requires high quality guidance from the property predictor, hence I retrained the predictor for *ESR1*-binding affinity with the full Zinc250k dataset. Using the trained predictor for guidance, 10,000 molecules were generated with the diffusion model and filtered for SA < 4, QED > 0.6 and no cycles with less than 5 or more than 6 atoms. The affinities of the generated molecules were then computed for verification using AutoDock-GPU [222]. The 3 top-scoring generated

molecules with respect to *ESR1*-affinity are shown in Figure 37. All 3 had an at least 2-times better dissociation constant (K_D) than the known antagonist Methylpiperidinopyrazole (MPP, $K_D \approx 4.72 * 10^{-7}$), for the top scoring generated molecule the K_D was almost 4-times better. Searching for the proposed ligands yielded no hits in the full ZINC database with its hundreds of millions of compounds [209,219], indicating their novelty. Further, the Tanimoto similarities [207] of the proposed ligands to the known antagonist MPP were computed and were 0.50, 0.47 and 0.41, sorted from highest to lowest ranking target affinity. To provide a point of reference, the Tanimoto similarity between MPP and the non-selective known *ESR1*-antagonist Fulvestrant is 0.45 and that between MPP and the endogenous ligand of *ESR1*, namely estradiol, is 0.42. Figure 37 B illustrates the top predicted ligand in the binding pocket of *ESR1*, figure 37 C additionally shows MPP. Illustrations of the poses of the second and third highest ranking proposed ligands can be found in supplementary figure 9.

The three potential ligands appear structurally very similar. I thus investigated whether there are distinct structural differences between molecules that the diffusion model generates when prompted for high affinity to *ESR1* and those when prompted for low affinity. For each scenario, I had the model generate 10,000 molecules, filtered for *SA* and *QED* as described above and selected the 10 top ranked hits for each scenario, i.e. high and low binding affinity (suppl. figure 10 A, B). The molecules generated for the two tasks show distinctly different structural attributes and molecular properties. The molecules generated for the high-affinity task have a generally higher number of atoms, including heavy atoms and subsequently a higher molecular weight, have a higher molar refractivity (polarisability of the compound), are more drug-like, have higher logP, penalised-logP and more rings. Their topological surface and formal charge is lower and they have less rotatable bonds as well as hydrogen-bond donors and receptors (suppl. figure 10 C).

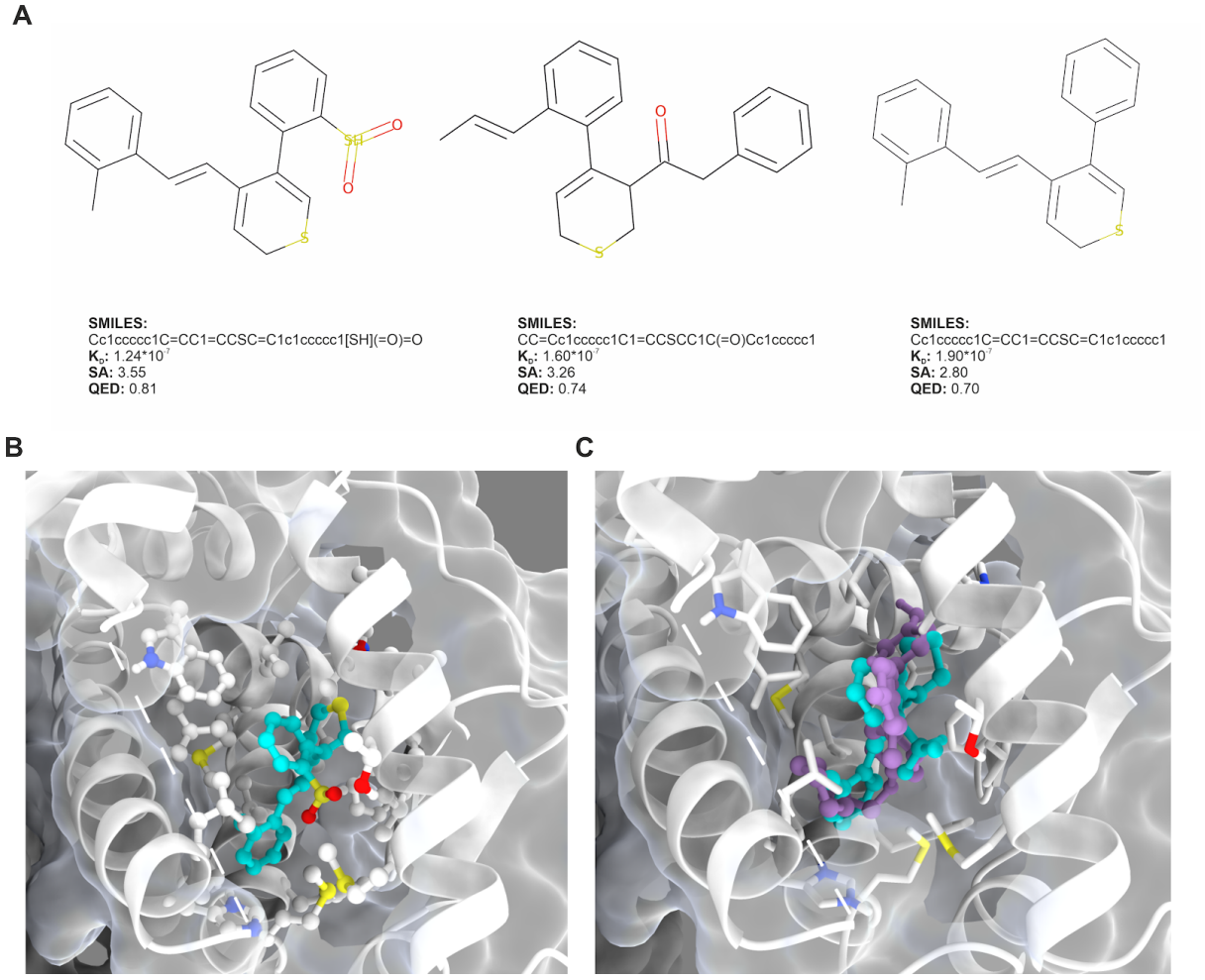


Figure 37: *ESR1*-ligand generation. **A**) Molecular graphs of the top three ligands generated by the diffusion model ranked according to their dissociation constant K_D as computed with AutoDock-GPU. Shown are also each molecule's SMILES string, synthetic accessibility (SA) and drug-likeness (QED); **B**) The first (best) generated ligand (turquoise) in its predicted binding pose in the binding pocket of *ESR1* (white) alone and **C)** with the known antagonist Methylpiperidinopyrazole in purple, visualised with ChimeraX.

Experiment 5: Gene-Expression Guidance

To increase the yield of drug trials, incorporating systemic biological effect-information into the development process is an appealing idea. However, such data is typically high dimensional and poses a much more difficult challenge than single-valued properties. To assess how our diffusion model responds to guidance towards highly complex properties, I included molecule-induced gene-expression as a target property. Here, I relied on a property predictor kindly provided by the authors of the *cGAN*-project [171].

I used a set of 3075 profiles from the L1000 dataset, that the predictor had not seen during training, to sample 100 molecules for each of the profiles. I used the predictor to guide the generation such that the generated molecules had a high predictive score for the corresponding profiles. I used five different guidance scales (5, 10, 25, 50 and 100),

representing increasing signal strength. As shown in Figure 38 A, the mean predicted scores for the generated molecules and their corresponding gene-expression profiles increases gradually with stronger guidance. Additionally, the mean scores with guidance are decisively higher than when the model is sampled without guidance. In fact, sampling without guidance mostly results in scores around 0 (suppl. figure 11), which is expected since the molecules are generated at random and are thus unlikely to match the profiles by chance. These results show very clearly that despite the complexity of the property, the model is responsive to the signal provided by the property predictor.

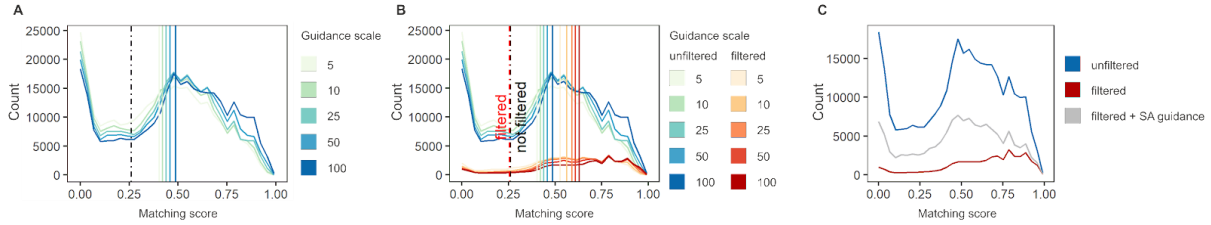


Figure 38: Gene-expression guidance. **A**) Predicted matching scores of molecules generated for given gene-expression profiles. Colours indicate different guidance scales used during generation, vertical lines indicate the mean predicted scores (the higher the better). The black dashed line is the mean predicted score of molecules generated without guidance, i.e. random molecules. **B**) Predicted matching scores of molecules generated for given gene-expression profiles without and with filtering for synthesisability (SA < 4.5) in blue shades and red shades, respectively. Colours additionally indicate guidance strength. Vertical lines represent mean predicted scores. The black and red dashed lines are the mean predicted scores of molecules generated without guidance, i.e. random molecules, without and with filtering for synthesisability. **C**) Predicted matching scores of molecules generated for given gene-expression profiles without filtering for synthesisability (blue), with filtering for synthesisability (red) and with filtering and additional guidance for synthesisability (grey), all using guidance strength 100.

Next, I filtered the generated molecules to only include molecules that are synthesisable (SA < 4.5). As illustrated in Figure 38 B, the overall amount of molecules left after the filter decreases drastically, as can be expected since the model did not receive any guidance with respect to synthesisability. Interestingly, the mean matching scores increase in comparison to the unfiltered molecules for all guidance scales, indicating that molecules that are difficult to synthesise are associated with a lower prediction score. Since the training of the diffusion model is independent of the property predictors used for guidance, more property predictors can be added during generation without the necessity of retraining. Therefore, I next added a second property predictor for synthesisability in addition to the gene-expression predictor to investigate if I could enrich the amount of molecules left after filtering for synthesisability. As demonstrated in Figure 38 C, this indeed manipulated the generative process to produce more synthesisable molecules.

I next compared the efficiency of our model to generate high scoring molecules for the given gene-expression profiles in comparison to the *cGAN* model. All following results presented for the diffusion model were retrieved using a guidance scale of 100. Since the *cGAN* model is SMILES-based, it struggles to generate valid molecules: approximately 10% of generated molecules are valid as stated by the authors [171]. Therefore, I sampled 10x as many molecules from the *cGAN*, i.e. 1000 molecules per gene-expression profile, than I sampled with the diffusion model, yielding a total of 3,075,000 molecules for the *cGAN*. After filtering these for validity and an $SA < 4.5$, a total of 51,477(~**2%**) molecules remained (see Table 5). Applying the same filter to our model sampled with only gene-expression guidance, 39,195 (~**13%**) of the total 307,500 molecules remained. When adding additional guidance for synthesisability, I could increase this yield to 131,652 (~**43%**). I then applied an additional cycle filter to the molecules sampled from both models: Only molecules that contained cycles with at least 5 and at most 6 atoms were kept. Molecules with smaller or larger cycles are often hard to synthesise but not efficiently detected by the *SA* score [174]. After applying this filter, 25,328(~**1%**) of molecules sampled from the *cGAN* and 106,774 (~**35%**) of the molecules sampled from our diffusion model remained. In a last step, I investigated the proportions of generated molecules with high predicted matching scores, i.e. those that best exhibit the desired property. I filtered for only those molecules that achieved a score of > 0.9. Although I used the distilled classifier for guidance, for this comparison I recomputed the scores of our generated molecules with the original classifier. This is to ensure that the comparison is fair and that superior results of either of the models are not due to a bias in the classifier. Thus, sampling for validity, $SA < 4.5$ and a gene-expression matching score > 0.9, the *cGAN* model was left with 27,936(~**1%**) and our diffusion model with 9,131(~**3%**) of the generated molecules, a 3-fold increase. Note that the difference in absolute number is due to the *cGAN* being sampled 10x more per gene-expression profile. Additionally adding the cycle filter, yielded 14,551(~**0.5%**) and 8,061 (~**2.6%**), for the *cGAN* and the diffusion model, respectively, a more than 5-fold increase for our model.

	cGAN			DM				
Condition (cGAN)/ guidance (DM)	gene-expression			gene-expression	gene-expression + SA	gene-expression	gene-expression + SA	gene-expression + SA
Filter	No filter	validity + SA	validity + SA + cycle	No filter	No filter	SA	SA	SA + cycle
Total #	3,075,000	51,477	25,328	307,500	307,500	39,195	131,652	106,774

molecules		(~2%)	(~1%)			(~13%)	(~43%)	(~35%)
Score > 0.9	NA*	27,936 **(~1%)**	14,551 **(~0.5%)**	11,302 (~4%)	13,411 (~4%)	5,522 (~2%)	9,131 **(~3%)**	8,061 **(~2.6%)**

Table 5: Number of molecules generated under different conditions and after applying various filters for the conditional GAN (cGAN) and the diffusion model (DM). Synthetic accessibility (SA) filter: SA < 4.5. Cycle filter: molecules with cycles of at least 5 and at most 6 atoms. Score: Predicted matching score between gene-expression profile and generated molecules. **NA***: Computation for invalid molecules is not possible.

Despite the good performance of the diffusion model, it must be noted that the predictions of the distilled predictor and that of the original predictor do not align very well in certain scenarios. While their predictions generally agree when comparing them for molecules sampled without guidance, they increasingly diverge for molecules generated with guidance.
Their agreement on a set of randomly sampled gene-expressions and molecules from the VAE latent space was already demonstrated above (for reference, see suppl. figure 7). When sampling a larger amount of molecules from the diffusion model without guidance, the results were noisier but the coefficient of determination was only slightly lower, with both predictors predicting mostly very low values (Figure 39 A). That is expected since the molecules are not sampled with gene-expression guidance. However, when sampling with i) gene-expression guidance and ii) gene-expression and *SA* guidance, the correspondence decreased drastically (Figure 39 B and C, respectively).
Particularly, the original predictor exhibits a noticeable bimodal behaviour when applying only gene-expression guidance, giving either very high or very low scores unexpectedly often. The least agreement can be observed when using both gene-expression and *SA* guidance.
Additionally, when applying the original predictor to a set of unknown true pairs of molecules and gene-expression profiles, the model returns an average score of less than 0.75, while in truth they should all be 1.

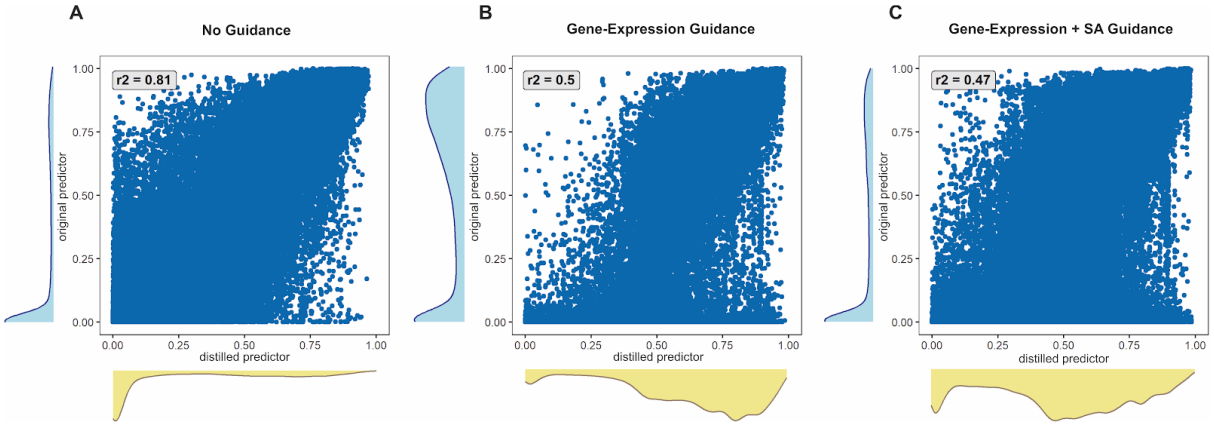


Figure 39: Agreement of the original and distilled molecule-gene-expression matching predictor. Shown are the scores predicted by the distilled (x-axis) and original (y-axis) predictor for **A**) molecules sampled from the diffusion model without guidance and randomly assigned gene-expression profiles, **B**) molecules sampled from the diffusion model with gene-expression guidance and corresponding gene-expression profiles, **C**) molecules sampled from the diffusion model with gene-expression guidance as well as synthesisability (SA) guidance and corresponding gene-expression profiles. Histograms show molecule density. The coefficient of determination (r^2) between the prediction of the two predictors is given in the top left corner of each panel.

4.4 Discussion

The objective of this project was to build a generative model that is capable of generating a diverse variety of molecules guided towards different molecular target properties. The latent diffusion model with classifier guidance presented here was able to generate a range of molecules that exhibited a high *novelty* ratio, i.e. no generation of molecules from the train set, *uniqueness*, i.e. not collapsing on a small set of molecules that it generates repeatedly and *diversity*, demonstrating that the model has gathered a vast understanding of chemical structures. The molecules generated in an unconditional setting without guidance covered the value range of the training dataset with respect to several molecular properties, showing that the model utilised the information contained in the train set to a large extent and did not only learn subsets of the training space. The experiments on unguided molecule generation thus illustrated that diffusion models are capable of successfully learning from molecular data for *de novo* generation of unseen molecules.

The additional experiments with guidance towards different target properties provided evidence that guided diffusion of molecules is possible without mode collapse. If generative models do not gather a comprehensive understanding of a domain, they collapse on one or

few modes that satisfy the imposed requirements and stick to only generating these modes. The latent diffusion model presented here, however, demonstrates good generalisation by generating a wide variety of different molecules that fit the requirements while maintaining a high internal diversity.

Moving beyond single-property guidance, also the multi-property setting provided good results, which is a key achievement for the model's application to real-world problems. Nonetheless, a point that needs further addressing in the context of multi-property guidance in the future is the selection of weights for the different properties. The signal strength that the diffusion model receives from the different property predictors varies, making it necessary to tune their signal with weights in order to give desired importance to the different properties. In an application setting, it is likely that the number of properties that are to be targeted increases further, and that the different properties have varying importance levels. While it was possible to empirically find suitable weights for the few property predictors used here, more work is required to automate this process to make it scalable for increasing numbers of controlled properties.

The model's performance in generating candidate ligands for the *ESR1* target looks very promising. The top scoring generated molecules show a much higher affinity to the target's binding pocket than a known antagonist, visual evaluation shows structural similarity between the proposed ligands, indicating an understanding of the model for a suitable ligand scaffold. Furthermore, the proposed ligands are estimated to be well synthesisable and drug-like. Their absence in the ZINC database comprising hundreds of millions of compounds shows that the diffusion model's capability to generate novel compounds goes far beyond the training set. The next mandatory step is experimental validation. Kits for running *ESR1* binding assays are commercially available, however, given that these molecules do not yet exist (at least not publicly, proprietary databases cannot be assessed), they need to be synthesised first with the additional expertise of medicinal chemists.

The experiments on guidance with gene-expression profiles showed that guided generation is not only possible with single-valued properties, but also with properties of high dimensionality. This is an important prospect with respect to using data from systems-medicine in the future, which is often high-dimensional. Interestingly, when guiding the generation to produce molecules that match given gene-expression profiles, but not using additional guidance for synthesisability, the matching scores increase when subsequently filtering out hard-to-synthesise molecules. This implies that these hard-to-synthesise molecules are associated with low matching scores by the property predictor. The reason for

this is likely a bias in the predictor training: The dataset used for training the property predictor was the L1000 dataset which documents the expression profiles induced by a variety of existing compounds. The proportion of difficult to synthesise molecules in this set can be expected to be low, biassing the predictor's predictive accuracy to such easily synthesisable compounds. The predicted low matching scores for hard-to-synthesise compounds must therefore be treated with care.

The reason for the disagreement between the original and the distilled predictor is not entirely clear, however, the most likely cause is that the distillation did not work properly. A more explicit incorporation of edge cases into the distillation may be helpful here. This is motivated by the observation that agreement between the two predictors is high when randomly sampling the molecular space but low when focusing on a more concise subspace as was the case with guidance. Such efforts and extensive computations would mostly be justified, if the original predictor was of very high quality. Unfortunately, applying the original predictor to a set of unknown true pairs of molecules and gene-expression profiles demonstrated its inability to correctly detect them as such. This is not entirely surprising because the amount of paired gene-expression and molecule data at least in the publicly accessible space is limited mostly to the L1000 dataset. In addition to the lack of data, the prediction task is complicated, given the high complexity of the gene-expression data.

Generally, the quality of the predictor is the key to success. The diffusion model adapts the generation to satisfy the requirements of the predictors, even if the predictor is wrong. To generate molecules of high quality, i.e. that truely exhibit the desired properties, good predictors are required. This observation concludes that the bottleneck of the approach presented in this project - i.e. the quality of the predictors - is therefore mitigated to a discriminative problem that is much easier to learn in comparison to a generative one. This automatically entails that the generative model itself (the diffusion model) only has to be trained once in an unconditional setting. Property-predictors of higher quality or for different properties can be trained separately and combined with the already trained diffusion model. This makes the presented approach resource friendly and highly customisable.

Besides the already mentioned points, future directions should certainly include substructure optimisation: Often during the drug development process, an identified lead compound is optimised towards the target property based on small structural changes. Passing a substructure to the model and fine-tuning it, rather than starting from scratch would therefore be a valuable addition to the model's capabilities.

Lastly, a serious issue that needs addressing is the dual-use problem of models like the one presented here. Generating molecules that manifest desired properties has many beneficial use cases, for example in the medical context or materials sciences. However, it also bears the inherent risk of being misused for malicious intent, for example to generate molecules that maximise toxicity. The model described in this work is solely intended to be used in a beneficial context and in accordance with the law. Future research directions to counteract malicious use of generative models specifically in the context of molecular generation could include developing generative models that, prompted with a dangerous substance, generate other molecules or aggregates of such that render such a substance inoperative.

5. Summary and Future Work

This thesis has expanded the current knowledge base of two research areas: Firstly, it has examined the potential of generative models in the field of systems medicine, specifically highlighting two application scenarios: 1) its utility for the generation of synthetic transcriptomic cohorts and additionally its ability to do so in a privacy-preserving way; 2) its applicability to property-based compound design using complex systems medicine properties such as drug-induced transcriptomics profiles and protein targets as well as other physicochemical target properties. Secondly, this work has addressed the topic of *green AI* and studied architecture optimisation to reduce resource consumption when training AI models.

Deep generative modelling has advanced rapidly in the past years, but its application has been mostly focused on image and text generation. However, the potential use-cases for generative models are vast and more diverse translational tasks must be explored. The biomedical field offers many opportunities to make use of this technology.
I addressed the topic of applying such models to construct synthetic patient cohorts of transcriptomics data in the ProGeneGen project. Together with the collaboration partners from CISPA, I demonstrated the utility of different model types: While the GAN training was too unstable to make it a reasonable choice, (DP-)MERF and (DP-)CVAE were viable options. DP-MERF, however, did not pass the first utility test by failing to generate a private synthetic cohort that allowed the training of an accurate classifier for real data. While the CVAE produced data with high enough utility for the classification task, both in the non-private and the private scenario, using synthetic cohorts instead of real cohorts in practice requires the data to also maintain key biological features. I therefore investigated the model's competence to create synthetic data that preserves differential expression of genes as well as the co-expression of genes under the studied conditions. This work demonstrated that the studied model is capable of preserving these features in a non-private generation setting without simply reconstructing the original data, but that this was not the case when the model was trained with differential privacy constraints. While this allows enrichment of existing cohorts with synthetic samples for internal use or for sharing with data sharing agreements, it does not yet allow the public release of private synthetic cohorts that maintain high utility. In this regard, more research is needed in the future. Concisely, future directions must include elaborating on why the utility of the private data is lost to such an extent. This prompts the

follow-up question of whether there is a tipping point with regard to the DP-parameters after which the utility of the data drops suddenly or if instead this is a gradual process. Another research direction is the incorporation of biology based utility metrics into the training process. The main focus in this regard must be on how to integrate these in a privacy-compliant way. Including domain knowledge rather than information based on the exact dataset at hand would be a promising additional approach, however, existing knowledge - especially in understudied diseases - is sparse and its documentation is not unified and often in free-form text, making it hard to extract. Lastly, the current trend in transcriptomics towards single-cell sequencing dictates that generative models for private cohort generation must also be explored in the context of single-cell data. The different statistical properties of single-cell data in comparison to data from bulk-sequencing will require special consideration. Two particular issues come to mind: Firstly, while privacy preservation in bulk data targets the protection of each data point (equivalent to a sample), in single-cell data, data points represent cells, many of which belong to the same sample. Thus, privacy protection must be guaranteed for a *group* of data points. Secondly, the high drop-out rate in single-cell data, i.e. zero-measurements that can either originate from the absence of the transcript or the failure to detect it, is often compensated for by imputation techniques that infer missing data points. If such techniques rely on other cells for their imputation, specifically, cells from other individuals, this requires extensive consideration in terms of the privacy budget.

In the DrugDiff project, generative modelling was addressed for the generation of new compounds based on desired target properties. This project had two main foci: 1) many types of generative models have been used for molecule generation in the past. Here, the focus was on the class of diffusion models which have recently had unparalleled success in the image domain; 2) In terms of resource management and energy consumption, a particular point of interest was the development of a model that allows conditioning but without the requirement of retraining the model whenever a condition is added or removed. Such an architecture comes with the additional benefit that it not only reduces computing resources but also makes the application of the model highly flexible by providing a plug-and-play mechanic.
With this project I have demonstrated that diffusion models are indeed applicable to the task of conditional molecule generation, particularly made possible by the use of latent diffusion models, without the need for retraining when adding new conditions. By training the diffusion model for the unconditional task of general molecule generation and outsourcing

the conditional setting to separate expert models in form of independently trained property predictors, I achieved that the generative model does not need retraining when adding further properties. Instead, predictors for such properties can be trained separately and then added to the existing diffusion model to exert guidance. I also demonstrated that the guided generation can be successfully applied to highly complex properties from systems medicine and biochemistry, such as induced gene-expression profiles and binding affinity to target proteins. I was able to validate generated ligands using docking simulations. Additionally, the proposed ligands were not found in public databases and demonstrated higher target affinity than known ligands.

There are several future directions that can be taken from here. Firstly, I trained the model on a database of small, drug-like molecules. Extensions of this model to other molecule domains are thinkable. These include for example macromolecules, such as antibodies or other proteins, or inorganic structures. Additionally, the application field could be expanded from drug development to also address questions from materials science or environmental science. This would require a different training set and also include different desired properties, but the concept can be expected to be translatable.

The performance of the diffusion model strongly depends on two factors: 1) the quality of the latent space that it learns from and 2) the competence of the property predictors that it turns to for guidance. Thus, improving the VAE used to create the latent space can further boost the diffusion model's performance. Concisely, here, more complex architectures such as transformers may be worth exploring. Regarding the latter point, better property predictors equate to better guidance. This poses a promising entrypoint for collaborations with the pharmaceutical industry, since the training of such predictors could profit dramatically from their already existing proprietary experimental data or their capability of producing such data sets.

To improve model performance, there is always the argument of larger training data sets. Here, I used a small set comprising only 250k molecules. However, as the second project of this thesis has demonstrated, more data does not always have to be the go-to solution, but that comparable improvements in performance can also be achieved by a) architecture optimisation and b) well crafted smaller datasets.

Another future direction could be the solution of the inverse problem: Instead of generating a compound, an interesting research question would be that *given* a compound, e.g. a toxin, to generate a small molecule, macromolecule or complex of molecules that inactivates or

absorbs the compound. Such research would counteract the addressed dual-use issue of the model presented here.
Lastly, another task would be to expand the target repertoire from proteins to also include the targeting of RNA.

Beyond the applications of generative modelling in systems medicine that were addressed here, the field offers further opportunities. As previously outlined, there exists a plethora of different omics data types that are collected in clinical research. Research into the augmentation and protection of these other data modalities should be an important future topic. Generally, the capability of generating synthetic patients in the context of various medical data types could help to tackle minority biases, be they ethnic minorities or patient groups suffering from rare diseases.

To successfully further research in this area, solutions for closer collaborations between computer scientists and natural scientists as well as physiologists are required. This particularly includes the expansion of positions that can communicate between the two fields and translate problems and proposed solutions. Such translators are important to avoid models being created to solve problems that are too far away from medical reality, be it because they over-simplify or misidentify the problem. They are also important to increase the acceptance of AI models in medical applications.

This thesis further addressed the issue of *green AI*. With AI models becoming an omnipresent part of our lives while we are facing a global crisis of continuous warming and dwindling resources, more must be done to contain the environmental impact of training AI. As demonstrated in the second project of this work, more research into *a priori* architecture optimisation is needed to move away from the idea that better models are automatically larger models. Such research could dramatically reduce the resources needed to train the models while maintaining a comparable performance. Additionally, understanding how architecture impacts model performance and being able to optimise this is likely to come with the byproduct of being able to better explain the way the models work.
Besides more research, there is also a required change in mentality. Oftentimes, mindful and planned-out programming is replaced by trial and error behaviour. This more often than not entails repeated running of code and training of models that do not result in a final product, while constantly consuming resources. While of course not all errors and issues can be anticipated, awareness for the consumed resources must be increased to change this mentality

as much as possible. A possible way could be the introduction of *green budgets* implemented by the providers of high performance computing (HPC) clusters: A report returned after running the compute job that translates the energy used into something tangible and less abstract, comparing it, for example, to the CO_2- footprint of a car.

Additionally, we need to move past the "battle for percentages". Many publications focus on solving already addressed problems with a different architecture reporting a performance increase of a few - or even fractions of - percent. Not only must it be considered, if the benefit of such small improvements justifies the resource consumption of the model, but also whether the reported improvement is indeed true: Minor decisions such as changing the seed or using a different GPU may cause such performance fluctuations without the presented architecture or method being the cause of the improvement.

First useful and progressive steps towards addressing these issues have already been taken by the research community, journals, conferences and HPC providers. For example, the HoreKa HPC at the Karlsruhe Institute for Technology uses a sophisticated cooling strategy that is based on warm water and the waste heat of the HPC is recycled for heating office spaces during the winter. It also reports the energy consumption of jobs back to the users. In addition to such innovative concepts, the topic of energy consumption by AI is increasingly addressed in publications and conferences and proposals are being made for reporting energy benchmarks alongside model performance. AI is aiding our everyday life in many aspects and it is inevitably going to play an even larger part in the future. Such promising developments towards green AI provide a positive outlook towards balancing the use and development of AI and the conservation of our environment in the years to come.

Altogether, generative modelling is continuously proving its contribution to the medical field. Its use for generating synthetic cohorts, protecting patient privacy and aiding the development of new drugs as presented in this thesis are topics that bear high beneficial potential. Future research in this matter as well as the expanded application of generative models to other aspects of medical research provide prospects of improved treatments and patient care.

6. Acknowledgements

I would like to thank Dr. Matthias Becker for the guidance and support he provided me with throughout the past years as his PhD student, for entrusting me with the freedom to ask and pursue my own research questions and for his help whenever I needed it.

I further want to express my thanks to Prof. Dr. Joachim L. Schultze for supporting me throughout the years, from my Bachelor and Master thesis all the way to my Dissertation and for reviewing this work as part of the dissertation committee.

I also want to emphasise my gratitude towards Prof. Dr. Alexander Effland for chairing the dissertation committee, Prof. Dr. Mario Fritz for reviewing this work and Priv.-Doz. Dr. Marc Beyer for agreeing to join my committee as the subject-external member.

I am very grateful to the entirety of the systems medicine department and particularly the friends I have made here. It is very rare to work in an environment where people come together from so many diverse areas of expertise, covering computer science, biology, medicine and everything in between. It has helped me to not lose focus of the application area of my work and I am extremely thankful for the many different things I have learned and continue to learn from the many different fields represented by the people around me.

I would also like to thank Erinc Merdivan and Marie Piraud from the Helmholtz AI consultant team for health. I learned a great deal when working with them on the realisation of the DrugDiff project for which I am very grateful.

Finally, I would like to thank my family. The few words I can write here are not nearly enough to say how grateful I am for their support. I want to express my gratitude towards my parents, for their trust and always supporting and always believing in me. And also towards my aunt for being so interested in my work and painting the beautiful images that illustrate the chapters of this thesis, as well as to my cousin, who has always been more like a brother to me. And lastly, I want to express my deepest appreciation to Sofía, for being so incredibly patient and supportive throughout this PhD adventure and without whom I would never have been able to do this.

7. Technical Note and Funding

This work was supported by the HGF Helmholtz AI grant Pro-Gene-Gen [ZT-I-PF5-23] as well as the Helmholtz Association's Initiative and Networking Fund on the HAICORE@FZJ partition.
The figures were partially created with Biorender.com.

8. References

1. Wolkenhauer, O., Auffray, C., Jaster, R., Steinhoff, G., and Dammann, O. (2013). The road from systems biology to systems medicine. Pediatr. Res. *73*, 502–507.
2. Apweiler, R., Beissbarth, T., Berthold, M.R., Blüthgen, N., Burmeister, Y., Dammann, O., Deutsch, A., Feuerhake, F., Franke, A., Hasenauer, J., *et al.* (2018). Whither systems medicine? Exp. Mol. Med. *50*, e453.
3. Palsson, B. (2002). In silico biology through “omics”. Nat. Biotechnol. *20*, 649–650.
4. Boright, A.P., Kere, J.A., and Scherer, S.W. (2003). Genomics and pediatric research. Pediatr. Res. *53*, 4–9.
5. Proteomics, transcriptomics: what’s in a name? (1999). Nature *402*, 715–715.
6. Corthals, G.L., and Nelson, P.S. (2001). Large-scale proteomics and its future impact on medicine. Pharmacogenomics J. *1*, 15–19.
7. Wenk, M.R. (2005). The emerging field of lipidomics. Nat. Rev. Drug Discov. *4*, 594–610.
8. Beck, S., Olek, A., and Walter, J. (1999). From genomics to epigenomics: a loftier view of life. Nat. Biotechnol. *17*, 1144.
9. Zambrowicz, B.P., and Sands, A.T. (2003). Knockouts model the 100 best-selling drugs--will they model the next 100? Nat. Rev. Drug Discov. *2*, 38–51.
10. Lu, C.Y., Terry, V., and Thomas, D.M. (2023). Precision medicine: affording the successes of science. NPJ Precis. Oncol. *7*, 3.
11. Gerhold, D.L., Jensen, R.V., and Gullans, S.R. (2002). Better therapeutics through microarrays. Nat. Genet. *32 Suppl*, 547–551.
12. Fasano, S., Milone, A., Nicoletti, G.F., Isenberg, D.A., and Ciccia, F. (2023). Precision medicine in systemic lupus erythematosus. Nat. Rev. Rheumatol.
13. Chang, Y., Deng, Q., and Bao, X. (2023). A pluripotent road to immunoengineering. Nat. Rev. Bioeng.
14. Bärthel, S., Falcomatà, C., Rad, R., Theis, F.J., and Saur, D. (2023). Single-cell profiling to explore pancreatic cancer heterogeneity, plasticity and response to therapy. Nat. Cancer *4*, 454–467.

15. Oestreich, M., Chen, D., Schultze, J.L., Fritz, M., and Becker, M. (2021). Privacy considerations for sharing genomics data. EXCLI J. *20*, 1243–1260.

16. Gürsoy, G., Li, T., Liu, S., Ni, E., Brannon, C.M., and Gerstein, M.B. (2022). Functional genomics data: privacy risk assessment and technological mitigation. Nat. Rev. Genet. *23*, 245–258.

17. Harmanci, A., and Gerstein, M. (2018). Analysis of sensitive information leakage in functional genomics signal profiles through genomic deletions. Nat. Commun. *9*, 2453.

18. Schadt, E.E., Woo, S., and Hao, K. (2012). Bayesian method to predict individual SNP genotypes from gene expression data. Nat. Genet. *44*, 603–608.

19. Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., and Aspuru-Guzik, A. (2018). Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent. Sci. *4*, 268–276.

20. Kusner, M.J., Paige, B., and Hernández-Lobato, J.M. (2017). Grammar Variational Autoencoder.pdf. Proceedings of the 34th International Conference on Machine Learning.

21. Lim, J., Ryu, S., Kim, J.W., and Kim, W.Y. (2018). Molecular generative model based on conditional variational autoencoder for de novo molecular design. J. Cheminform. *10*, 31.

22. Schwartz, R., Dodge, J., Smith, N.A., and Etzioni, O. (2019). Green AI. arXiv.

23. Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Stroudsburg, PA, USA: Association for Computational Linguistics), pp. 3645–3650.

24. Verdecchia, R., Sallou, J., and Cruz, L. (2023). A Systematic Review of Green AI. arXiv.org.

25. Sohl-Dickstein, J. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics . arXiv.

26. Song, Y., and Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. Advances in Neural Information Processing Systems.

27. Ho, J. (2020). Denoising Diffusion Probabilistic Models. arXiv.

28. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. arXiv.

29. Wouters, O.J., McKee, M., and Luyten, J. (2020). Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018. JAMA *323*, 844–853.

30. Sun, D., Gao, W., Hu, H., and Zhou, S. (2022). Why 90% of clinical drug development fails and how to improve it? Acta Pharm. Sin. B *12*, 3049–3062.

31. Hingorani, A.D., Kuan, V., Finan, C., Kruger, F.A., Gaulton, A., Chopade, S., Sofat, R., MacAllister, R.J., Overington, J.P., Hemingway, H., *et al.* (2019). Improving the odds of drug development success through human genomics: modelling study. Sci. Rep. *9*, 18911.

32. Paul, D., Sanap, G., Shenoy, S., Kalyane, D., Kalia, K., and Tekade, R.K. (2021). Artificial intelligence in drug discovery and development. Drug Discov. Today *26*, 80–93.

33. Sellwood, M.A., Ahmed, M., Segler, M.H., and Brown, N. (2018). Artificial intelligence in drug discovery. Future Med. Chem. *10*, 2025–2028.

34. Fleming, N. (2018). How artificial intelligence is changing drug discovery. Nature *557*, S55–S57.

35. Neary, B., Zhou, J., and Qiu, P. (2021). Identifying gene expression patterns associated with drug-specific survival in cancer patients. Sci. Rep. *11*, 5004.

36. Costello, J.C., Heiser, L.M., Georgii, E., Gönen, M., Menden, M.P., Wang, N.J., Bansal, M., Ammad-ud-din, M., Hintsanen, P., Khan, S.A., *et al.* (2014). A community effort to assess and improve drug sensitivity prediction algorithms. Nat. Biotechnol. *32*, 1202–1212.

37. Chengalvala, M.V., Chennathukuzhi, V.M., Johnston, D.S., Stevis, P.E., and Kopf, G.S. (2007). Gene expression profiling and its practice in drug development. Curr. Genomics *8*, 262–270.

38. MotieGhader, H., Safavi, E., Rezapour, A., Amoodizaj, F.F., and Iranifam, R.A. (2021). Drug repurposing for coronavirus (SARS-CoV-2) based on gene co-expression network analysis. Sci. Rep. *11*, 21872.

39. Chawla, S., Rockstroh, A., Lehman, M., Ratther, E., Jain, A., Anand, A., Gupta, A., Bhattacharya, N., Poonia, S., Rai, P., *et al.* (2022). Gene expression based inference of cancer drug sensitivity. Nat. Commun. *13*, 5680.

40. Wu, P., Feng, Q., Kerchberger, V.E., Nelson, S.D., Chen, Q., Li, B., Edwards, T.L., Cox, N.J., Phillips, E.J., Stein, C.M., *et al.* (2022). Integrating gene expression and clinical data to identify drug repurposing candidates for hyperlipidemia and hypertension. Nat. Commun. *13*, 46.

41. Went, M., Hoang, P.H., Law, P.J., Kaiser, M.F., and Houlston, R.S. (2022). Exploiting gene dependency to inform drug development for multiple myeloma. Sci. Rep. *12*, 12696.

42. Ulusoy, I., and Bishop, C.M. (2006). Comparison of generative and discriminative techniques for object detection and classification. In Toward Category-Level Object Recognition Lecture notes in computer science., J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, eds. (Berlin, Heidelberg: Springer Berlin Heidelberg), pp. 173–195.

43. Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C.G. (2022). Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows,

Energy-Based and Autoregressive Models. IEEE Trans. Pattern Anal. Mach. Intell. *44*, 7327–7347.

44. Berisha, V., Krantsevich, C., Hahn, P.R., Hahn, S., Dasarathy, G., Turaga, P., and Liss, J. (2021). Digital medicine and the curse of dimensionality. npj Digital Med. *4*, 153.
45. Alaa, A., van Breugel, B., Saveliev, E., and van der Schaar, M. (2021). How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models. arXiv.org.
46. Kingma, D.P., and Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv.
47. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. NIPS.
48. Liu, F., and Demosthenes, P. (2022). Real-world data: a brief review of the methods, applications, challenges and opportunities. BMC Med. Res. Methodol. *22*, 287.
49. Tu, J.V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. J. Clin. Epidemiol. *49*, 1225–1231.
50. Schmidhuber, J. (2015). Deep learning in neural networks: an overview. Neural Netw. *61*, 85–117.
51. McCulloch, W.S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. *5*, 115–133.
52. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. *65*, 386–408.
53. Strang, G. (2019). VII: Learning from Data. In Linear Algebra and Learning from Data , p. 372.
54. Strang, G. (2019). Deep Learning and Neural Nets. In Linear Algebra and Learning from Data, p. iii.
55. Strang, G. (2019). VII.1 The Construction of Deep Neural Networks. In Linear Algebra and Learning from Data, p. 383.
56. Strang, G. (2019). VII: Hyperparameters: The Fateful Decisions. In Linear Algebra and Learning from Data, p. 411.
57. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. Nature *323*, 533–536.
58. Jordan, M.I. (1986). Serial Order: A Parallel Distributed Processing Approach (University of California San Diego).
59. Elman, J.L. (1990). Finding Structure in Time. Cogn. Sci. *14*, 179–211.
60. Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen.
61. Graves, A., Mohamed, A., and Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. arXiv.

62. Zhao, B., Li, X., and Lu, X. (2019). Hierarchical Recurrent Neural Network for Video Summarization. arXiv.

63. Pan, P., Xu, Z., Yang, Y., Wu, F., and Zhuang, Y. (2015). Hierarchical Recurrent Neural Encoder for Video Representation with Application to Captioning. arXiv.

64. Nah, S., Son, S., and Lee, K.M. (2019). Recurrent Neural Networks With Intra-Frame Iterations for Video Deblurring. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (IEEE), pp. 8094–8103.

65. Sutskever, I., Vinyals, O., and Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. NIPS.

66. Hewage, P., Behera, A., Trovati, M., and Pereira, E. (2019). Long-Short Term Memory for an Effective Short-Term Weather Forecasting Model Using Surface Weather Data. In Artificial intelligence applications and innovations: 15th IFIP WG 12.5 international conference, AIAI 2019, hersonissos, crete, greece, may 24–26, 2019, proceedings IFIP advances in information and communication technology., J. MacIntyre, I. Maglogiannis, L. Iliadis, and E. Pimenidis, eds. (Cham: Springer International Publishing), pp. 382–390.

67. Sherstinsky, A. (2018). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. arXiv.org.

68. Lipton, Z.C., Berkowitz, J., and Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv.

69. Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural Comput. *9*, 1735–1780.

70. Gers, F.A., and Schmidhuber, J. (2000). Recurrent nets that time and count. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (IEEE), pp. 189–194 vol.3.

71. Yao, K., Cohn, T., Vylomova, K., Duh, K., and Dyer, C. (2015). Depth-Gated LSTM. arXiv.

72. Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A Clockwork RNN. arXiv.

73. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Stroudsburg, PA, USA: Association for Computational Linguistics), pp. 1724–1734.

74. Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv.

75. Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv.

76. Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv.

77. Bahuleyan, H., Mou, L., Vechtomova, O., and Poupart, P. (2017). Variational Attention for Sequence-to-Sequence Models. arXiv.

78. Dollar, O., Joshi, N., Beck, D.A.C., and Pfaendtner, J. (2021). Attention-based generative models for de novo molecular design. Chem. Sci. *12*, 8362–8372.

79. Kingma, D.P., and Welling, M. (2013). Auto-encoding variational bayes. arXiv.

80. Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2016). On the Quantitative Analysis of Decoder-Based Generative Models. arXiv.

81. Song, Y. (2021). Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song. Available at: https://yang-song.net/blog/2021/score/ [Accessed October 27, 2022].

82. Pratella, D., Ait-El-Mkadem Saadi, S., Bannwarth, S., Paquis-Fluckinger, V., and Bottini, S. (2021). A survey of autoencoder algorithms to pave the diagnosis of rare diseases. Int. J. Mol. Sci. *22*.

83. Eiteneuer, B., Hranisavljevic, N., and Niggemann, O. (2019). Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder. In 2019 IEEE International Conference on Industrial Technology (ICIT) (IEEE), pp. 1286–1292.

84. Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. Philosophical Magazine Series 6 *2*, 559–572.

85. Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. *24*, 417–441.

86. Siwek, K., and Osowski, S. (2017). Autoencoder versus PCA in face recognition. In 2017 18th International Conference on Computational Problems of Electrical Engineering (CPEE) (IEEE), pp. 1–4.

87. Kullback, S., and Leibler, R.A. (1951). On information and sufficiency. Ann. Math. Statist. *22*, 79–86.

88. Sohn, K., Yan, X., and Lee, H. (2015). Learning Structured Output Representation using Deep Conditional Generative Models . NIPS 2015.

89. Mohamed, S., and Lakshminarayanan, B. (2016). Learning in Implicit Generative Models. arXiv.

90. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. arXiv.

91. Mirza, M., and Osindero, S. (2014). Conditional Generative Adversarial Nets. arXiv.

92. Weng, L. (2021). What are Diffusion Models? | Lil'Log. Available at: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/ [Accessed October 27, 2022].

93. Nichol, A., and Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic

Models . Proceedings of the 38 th International Conference on Machine Learning.

94. Hyvarinen, A. (2005). Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research.

95. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-Based Generative Modeling through Stochastic Differential Equations. arXiv.

96. Song, J., Meng, C., and Ermon, S. (2021). Denoising Diffusion Implicit Models. ICLR.

97. Dhariwal, P., and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. arXiv.

98. Crick, F. (1970). Central dogma of molecular biology. Nature *227*, 561–563.

99. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2002). Analyzing Protein Structure and Function.

100. Cramer, P., Bushnell, D.A., and Kornberg, R.D. (2001). Structural basis of transcription: RNA polymerase II at 2.8 angstrom resolution. Science *292*, 1863–1876.

101. Girbig, M., Misiaszek, A.D., and Müller, C.W. (2022). Structural insights into nuclear transcription by eukaryotic DNA-dependent RNA polymerases. Nat. Rev. Mol. Cell Biol. *23*, 603–622.

102. transcriptome | Learn Science at Scitable Available at: https://www.nature.com/scitable/definition/transcriptome-296/ [Accessed April 12, 2023].

103. Lockhart, D.J., and Winzeler, E.A. (2000). Genomics, gene expression and DNA arrays. Nature *405*, 827–836.

104. Wang, D., and Farhana, A. (2023). Biochemistry, RNA Structure. In StatPearls (Treasure Island (FL): StatPearls Publishing).

105. Zeng, Y. (2006). Principles of micro-RNA production and maturation. Oncogene *25*, 6156–6162.

106. Mattick, J.S., Amaral, P.P., Carninci, P., Carpenter, S., Chang, H.Y., Chen, L.-L., Chen, R., Dean, C., Dinger, M.E., Fitzgerald, K.A., *et al.* (2023). Long non-coding RNAs: definitions, functions, challenges and recommendations. Nat. Rev. Mol. Cell Biol. *24*, 430–447.

107. National Human Genome Research Institute Home | NHGRI Available at: https://www.genome.gov/ [Accessed May 16, 2023].

108. Casamassimi, A., and Ciccodicola, A. (2019). Transcriptional regulation: molecules, involved mechanisms, and misregulation. Int. J. Mol. Sci. *20*.

109. Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. Nat. Rev. Genet. *10*, 57–63.

110. Bai, J.P.F., Alekseyenko, A.V., Statnikov, A., Wang, I.-M., and Wong, P.H. (2013). Strategic applications of gene expression: from drug discovery/development to bedside.

AAPS J. *15*, 427–437.

111. Mardis, E.R. (2017). DNA sequencing technologies: 2006-2016. Nat. Protoc. *12*, 213–218.

112. Metzker, M.L. (2010). Sequencing technologies - the next generation. Nat. Rev. Genet. *11*, 31–46.

113. Goodwin, S., McPherson, J.D., and McCombie, W.R. (2016). Coming of age: ten years of next-generation sequencing technologies. Nat. Rev. Genet. *17*, 333–351.

114. Illumina An Introduction to Next-Generation Sequencing Technology. Available at: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf [Accessed April 12, 2023].

115. Houlihan, G., Arangundy-Franklin, S., Porebski, B.T., Subramanian, N., Taylor, A.I., and Holliger, P. (2020). Discovery and evolution of RNA and XNA reverse transcriptase function and fidelity. Nat. Chem. *12*, 683–690.

116. Li, X., and Wang, C.-Y. (2021). From bulk, single-cell to spatial RNA sequencing. Int. J. Oral Sci. *13*, 36.

117. Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B.B., Siddiqui, A., *et al.* (2009). mRNA-Seq whole-transcriptome analysis of a single cell. Nat. Methods *6*, 377–382.

118. Salem, H., Attiya, G., and El-Fishawy, N. (2017). Classification of human cancer diseases by gene expression profiles. Appl. Soft Comput. *50*, 124–134.

119. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., *et al.* (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science *286*, 531–537.

120. Gruvberger, S., Ringnér, M., Chen, Y., Panavally, S., Saal, L.H., Borg A, Fernö, M., Peterson, C., and Meltzer, P.S. (2001). Estrogen receptor status in breast cancer is associated with remarkably distinct gene expression patterns. Cancer Res. *61*, 5979–5984.

121. Supplitt, S., Karpinski, P., Sasiadek, M., and Laczmanska, I. (2021). Current achievements and applications of transcriptomics in personalized cancer medicine. Int. J. Mol. Sci. *22*.

122. Stebbing, J., and Singh Wasan, H. (2019). Decoding metastatic colorectal cancer to improve clinical decision making. J. Clin. Oncol. *37*, 1847–1850.

123. Reis-Filho, J.S., and Pusztai, L. (2011). Gene expression profiling in breast cancer: classification, prognostication, and prediction. Lancet *378*, 1812–1823.

124. Xing, S., Zheng, X., Wei, L.-Q., Song, S.-J., Liu, D., Xue, N., Liu, X.-M., Wu, M.-T., Zhong, Q., Huang, C.-M., *et al.* (2017). Development and Validation of a Serum Biomarker Panel for the Detection of Esophageal Squamous Cell Carcinoma through RNA Transcriptome Sequencing. J. Cancer *8*, 2346–2355.

125. Mortazavi, A., Williams, B.A., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nat. Methods *5*, 621–628.

126. Wilkerson, D.G., Crowell, C.R., Carlson, C.H., McMullen, P.W., Smart, C.D., and Smart, L.B. (2022). Comparative transcriptomics and eQTL mapping of response to Melampsora americana in selected Salix purpurea F2 progeny. BMC Genomics *23*, 71.

127. Wirka, R.C., Pjanic, M., and Quertermous, T. (2018). Advances in transcriptomics: investigating cardiovascular disease at unprecedented resolution. Circ. Res. *122*, 1200–1220.

128. Cummings, B.B., Marshall, J.L., Tukiainen, T., Lek, M., Donkervoort, S., Foley, A.R., Bolduc, V., Waddell, L.B., Sandaradura, S.A., O'Grady, G.L., *et al.* (2017). Improving genetic diagnosis in Mendelian disease with transcriptome sequencing. Sci. Transl. Med. *9*.

129. Huang, J., Galal, G., Etemadi, M., and Vaidyanathan, M. (2022). Evaluation and mitigation of racial bias in clinical machine learning models: scoping review. JMIR Med. Inform. *10*, e36388.

130. Brittain, H.K., Scott, R., and Thomas, E. (2017). The rise of the genome and personalised medicine. Clin. Med. *17*, 545–551.

131. The White House (2015). Precision Medicine Initiative | The White House. Available at: https://obamawhitehouse.archives.gov/precision-medicine [Accessed June 19, 2021].

132. Bernier, A., Liu, H., and Knoppers, B.M. (2021). Computational tools for genomic data de-identification: facilitating data protection law compliance. Nat. Commun. *12*, 6949.

133. Lin, Z., Owen, A.B., and Altman, R.B. (2004). Genomic research and human subject privacy. Science *305*, 183.

134. Huijsmans, R., Damen, J., van der Linden, H., and Hermans, M. (2007). Single nucleotide polymorphism profiling assay to confirm the identity of human tissues. J. Mol. Diagn. *9*, 205–213.

135. El Emam, K. (2011). Methods for the de-identification of electronic health records for genomic research. Genome Med. *3*, 25.

136. EUR-Lex - 32000D0520 - EN Available at: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32000D0520:EN:HTML [Accessed April 14, 2023].

137. Requirements of Participation | Privacy Shield Available at: https://www.privacyshield.gov/article?id=Requirements-of-Participation [Accessed April 14, 2023].

138. Case C-362/14 Maximillian Schrems v Data Protection Commissioner Available at: https://curia.europa.eu/juris/document/document.jsf;jsessionid=DD50484BEB7CB9B7449A16FDA04F0E90?text=&docid=169195&pageIndex=0&doclang=EN&mode=lst&

dir=&occ=first&part=1&cid=5251302 [Accessed May 6, 2023].

139. Case C-311/18 Data Protection Commissioner v Facebook Ireland Limited and Maximillian Schrems Available at: https://curia.europa.eu/juris/document/document.jsf?text=&docid=228677&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=5251438 [Accessed May 6, 2023].

140. Methods for De-identification of PHI | HHS.gov Available at: https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html [Accessed April 14, 2023].

141. Gürsoy, G., Emani, P., Brannon, C.M., Jolanki, O.A., Harmanci, A., Strattan, J.S., Cherry, J.M., Miranker, A.D., and Gerstein, M. (2020). Data Sanitization to Reduce Private Information Leakage from Functional Genomics. Cell *183*, 905-917.e16.

142. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., and Arcas, B.A. ̈uera y (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. AISTATS.

143. Warnat-Herresthal, S., Schultze, H., Shastry, K.L., Manamohan, S., Mukherjee, S., Garg, V., Sarveswara, R., Händler, K., Pickkers, P., Aziz, N.A., *et al.* (2021). Swarm Learning for decentralized and confidential clinical machine learning. Nature *594*, 265–270.

144. Gootjes-Dreesbach, L., Sood, M., Sahay, A., Hofmann-Apitius, M., and Fröhlich, H. (2020). Variational autoencoder modular bayesian networks for simulation of heterogeneous clinical study data. Front. Big Data *3*, 16.

145. Sood, M., Sahay, A., Karki, R., Emon, M.A., Vrooman, H., Hofmann-Apitius, M., and Fröhlich, H. (2020). Realistic simulation of virtual multi-scale, multi-modal patient trajectories using Bayesian networks and sparse auto-encoders. Sci. Rep. *10*, 10971.

146. Savage, N. (2023). Synthetic data could be better than real data. Nature.

147. Food and Drug Administration, U.S. Department of Health and Human Services (2004). Challenge and Opportunity on the Critical Path to New Medical Products (Food and Drug Administration, U.S. Department of Health and Human Services).

148. Lamberti, M.J., Wilkinson, M., Donzanti, B.A., Wohlhieter, G.E., Parikh, S., Wilkins, R.G., and Getz, K. (2019). A study on the application and use of artificial intelligence to support drug development. Clin. Ther. *41*, 1414–1426.

149. Lind, A.P., and Anderson, P.C. (2019). Predicting drug activity against cancer cells by random forest models based on minimal genomic information and chemical properties. PLoS ONE *14*, e0219774.

150. Beck, J.T., Rammage, M., Jackson, G.P., Preininger, A.M., Dankwa-Mullan, I., Roebuck, M.C., Torres, A., Holtzen, H., Coverdill, S.E., Williamson, M.P., *et al.* (2020). Artificial intelligence tool for optimizing eligibility screening for clinical trials in a large community cancer center. JCO Clin. Cancer Inform. *4*, 50–59.

151. Smalley, E. (2017). AI-powered drug discovery captures pharma interest. Nat. Biotechnol. *35*, 604–605.

152. Lee, W.-Y., Lee, C.-Y., and Kim, C.-E. (2023). Predicting activatory and inhibitory drug-target interactions based on structural compound representations and genetically perturbed transcriptomes. PLoS ONE *18*, e0282042.

153. Sadybekov, A.V., and Katritch, V. (2023). Computational approaches streamlining drug discovery. Nature *616*, 673–685.

154. A.I. Is Everywhere and Evolving - The New York Times Available at: https://www.nytimes.com/2021/02/23/technology/ai-innovation-privacy-seniors-education.html [Accessed April 16, 2023].

155. Artificial Intelligence: On a mission to Make Clinical Drug Development Faster and Smarter | Pfizer Available at: https://www.pfizer.com/news/articles/artificial_intelligence_on_a_mission_to_make_clinical_drug_development_faster_and_smarter [Accessed April 16, 2023].

156. Artificial intelligence (AI) in drug discovery - Research | Merck Available at: https://www.merckgroup.com/en/research/science-space/envisioning-tomorrow/precision-medicine/generativeai.html [Accessed April 16, 2023].

157. AI in Pharma | Bayer Global Available at: https://www.bayer.com/en/pharma/ai-pharma [Accessed April 16, 2023].

158. How AI could revolutionize drug discovery | McKinsey Available at: https://www.mckinsey.com/industries/life-sciences/our-insights/how-ai-could-revolutionize-drug-discovery [Accessed April 16, 2023].

159. Adopting AI in Drug Discovery | BCG Available at: https://www.bcg.com/publications/2022/adopting-ai-in-pharmaceutical-discovery [Accessed April 16, 2023].

160. Mullard, A. (2017). The drug-maker's guide to the galaxy. Nature *549*, 445–447.

161. Polishchuk, P.G., Madzhidov, T.I., and Varnek, A. (2013). Estimation of the size of drug-like chemical space based on GDB-17 data. J. Comput. Aided Mol. Des. *27*, 675–679.

162. Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J. Chem. Inf. Model. *28*, 31–36.

163. O'Boyle, N.M. (2012). Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. J. Cheminform. *4*, 22.

164. Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. (2020). Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. Mach. Learn.: Sci. Technol. *1*, 045024.

165. Rogers, D., and Hahn, M. (2010). Extended-connectivity fingerprints. J. Chem. Inf. Model. *50*, 742–754.

166. Durant, J.L., Leland, B.A., Henry, D.R., and Nourse, J.G. (2002). Reoptimization of MDL keys for use in drug discovery. J. Chem. Inf. Comput. Sci. *42*, 1273–1280.

167. Méndez-Lucio, O., Nicolaou, C., and Earnshaw, B. (2022). MolE: a molecular foundation model for drug discovery. arXiv.

168. David, L., Thakkar, A., Mercado, R., and Engkvist, O. (2020). Molecular representations in AI-driven drug discovery: a review and practical guide. J. Cheminform. *12*, 56.

169. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. IEEE Trans. Neural Netw. *20*, 61–80.

170. Kuzminykh, D., Polykovskiy, D., Kadurin, A., Zhebrak, A., Baskov, I., Nikolenko, S., Shayakhmetov, R., and Zhavoronkov, A. (2018). 3D molecular representations based on the wave transform for convolutional neural networks. Mol. Pharm. *15*, 4378–4385.

171. Méndez-Lucio, O., Baillif, B., Clevert, D.-A., Rouquié, D., and Wichard, J. (2020). De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. Nat. Commun. *11*, 10.

172. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). Learning Deep Generative Models of Graphs. arXiv.

173. Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction Tree Variational Autoencoder for Molecular Graph Generation . Proceedings of the 35th International Conference on Machine Learning.

174. Eckmann, P., Sun, K., Zhao, B., Feng, M., Gilson, M.K., and Yu, R. (2022). LIMO: Latent Inceptionism for Targeted Molecule Generation. In Proceedings of the 39th International Conference on Machine Learning (PMLR).

175. Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. (2020). GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. arXiv.

176. Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. (2018). Syntax-Directed Variational Autoencoder for Structured Data. arXiv.

177. Pham, T.-H., Xie, L., and Zhang, P. (2022). FAME: Fragment-based Conditional Molecular Generation for Phenotypic Drug Discovery. BioRxiv.

178. Born, J., Manica, M., Oskooei, A., Cadow, J., Markert, G., and Rodríguez Martínez, M. (2021). PaccMannRL: De novo generation of hit-like anticancer molecules from transcriptomic data via reinforcement learning. iScience *24*, 102269.

179. Shayakhmetov, R., Kuznetsov, M., Zhebrak, A., Kadurin, A., Nikolenko, S., Aliper, A., and Polykovskiy, D. (2020). Molecular generation for desired transcriptome changes with adversarial autoencoders. Front. Pharmacol. *11*, 269.

180. Zhavoronkov, A., Ivanenkov, Y.A., Aliper, A., Veselov, M.S., Aladinskiy, V.A., Aladinskaya, A.V., Terentiev, V.A., Polykovskiy, D.A., Kuznetsov, M.D., Asadulaev, A., *et al.* (2019). Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nat. Biotechnol. *37*, 1038–1040.

181. Westermayr, J., Gilkes, J., Barrett, R., and Maurer, R.J. (2023). High-throughput property-driven generative design of functional organic molecules. Nat. Comput. Sci.

182. Data sharing in the age of deep learning. (2023). Nat. Biotechnol. *41*, 433.

183. Marouf, M., Machart, P., Bansal, V., Kilian, C., Magruder, D.S., Krebs, C.F., and Bonn, S. (2020). Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks. Nat. Commun. *11*, 166.

184. Treppner, M., Salas-Bastos, A., Hess, M., Lenz, S., Vogel, T., and Binder, H. (2021). Synthetic single cell RNA sequencing data from small pilot studies using deep generative models. Sci. Rep. *11*, 9403.

185. Lall, S., Ray, S., and Bandyopadhyay, S. (2022). LSH-GAN enables in-silico generation of cells for small sample high dimensional scRNA-seq data. Commun. Biol. *5*, 577.

186. Pandey, D., and Onkara, P.P. (2023). Improved downstream functional analysis of single-cell RNA-sequence data using DGAN. Sci. Rep. *13*, 1618.

187. Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography Lecture notes in computer science., S. Halevi and T. Rabin, eds. (Berlin, Heidelberg: Springer Berlin Heidelberg), pp. 265–284.

188. Dwork, C. (2011). The promise of differential privacy: A tutorial on algorithmic techniques. In 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (IEEE), pp. 1–2.

189. Dwork, C., and Roth, A. (2014). The algorithmic foundations of differential privacy. FNT in Theoretical Computer Science *9*, 211–407.

190. Anders, S., and Huber, W. (2010). Differential expression analysis for sequence count data. Genome Biol. *11*, R106.

191. Costa-Silva, J., Domingues, D., and Lopes, F.M. (2017). RNA-Seq differential expression analysis: An extended review and a software tool. PLoS ONE *12*, e0190152.

192. Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., and Smyth, G.K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. *43*, e47.

193. Robinson, M.D., McCarthy, D.J., and Smyth, G.K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics *26*, 139–140.

194. Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin *1*, 80.

195. Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. *15*, 550.

196. de Torrenté, L., Zimmerman, S., Suzuki, M., Christopeit, M., Greally, J.M., and Mar, J.C. (2020). The shape of gene expression distributions matter: how incorporating distribution shape improves the interpretation of cancer transcriptomic data. BMC Bioinformatics *21*, 562.

197. Oestreich, M., Holsten, L., Agrawal, S., Dahm, K., Koch, P., Jin, H., Becker, M., and Ulas, T. (2022). hCoCena: horizontal integration and analysis of transcriptomics datasets. Bioinformatics *38*, 4727–4734.

198. Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. arXiv.com.

199. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16 (New York, New York, USA: ACM Press), pp. 308–318.

200. Harder, F., Adamczewski, K., and Park, M. (2020). DP-MERF: Differentially Private Mean Embeddings with Random Features for Practical Privacy-Preserving Data Generation. arXiv.

201. Warnat-Herresthal, S., Perrakis, K., Taschler, B., Becker, M., Baßler, K., Beyer, M., Günther, P., Schulte-Schrepping, J., Seep, L., Klee, K., *et al.* (2020). Scalable Prediction of Acute Myeloid Leukemia Using High-Dimensional Machine Learning and Blood Transcriptomics. iScience *23*, 100780.

202. Subramanian, A., Narayan, R., Corsello, S.M., Peck, D.D., Natoli, T.E., Lu, X., Gould, J., Davis, J.F., Tubelli, A.A., Asiedu, J.K., *et al.* (2017). A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. Cell *171*, 1437-1452.e17.

203. Sattarov, B., Baskin, I.I., Horvath, D., Marcou, G., Bjerrum, E.J., and Varnek, A. (2019). De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. J. Chem. Inf. Model. *59*, 1182–1196.

204. Bjerrum, E.J., and Sattarov, B. (2018). Improving Chemical Autoencoder Latent Space and Molecular De Novo Generation Diversity with Heteroencoders. Biomolecules *8*.

205. Prykhodko, O., Johansson, S.V., Kotsias, P.-C., Arús-Pous, J., Bjerrum, E.J., Engkvist, O., and Chen, H. (2019). A de novo molecular generation method using latent vector based generative adversarial network. J. Cheminform. *11*, 74.

206. Williams, R.J., and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. Neural Comput. *1*, 270–280.

207. Tanimoto, T.T. (1958). An elementary mathematical theory of classification and prediction (International Business Machines (IBM) Corporation).

208. Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., *et al.* (2020). Molecular sets (MOSES): A benchmarking platform for molecular generation models. Front. Pharmacol. *11*, 565644.

209. Sterling, T., and Irwin, J.J. (2015). ZINC 15--Ligand Discovery for Everyone. J. Chem. Inf. Model. *55*, 2324–2337.

210. Wang, R., Fu, Y., and Lai, L. (1997). A New Atom-Additive Method for Calculating Partition Coefficients. J. Chem. Inf. Comput. Sci. *37*, 615–621.

211. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR.

212. Deloitte pharma study: Drop-off in returns on R&D investments – sharp decline in peak sales per asset Available at: https://www2.deloitte.com/ch/en/pages/press-releases/articles/deloitte-pharma-study-drop-off-in-returns-on-r-and-d-investments-sharp-decline-in-peak-sales-per-asset.html [Accessed May 8, 2023].

213. Deloitte (2023). Seize the digital momentum Measuring the return from pharmaceutical innovation 2022.

214. Kadurin, A., Nikolenko, S., Khrabrov, K., Aliper, A., and Zhavoronkov, A. (2017). druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico. Mol. Pharm. *14*, 3098–3104.

215. Simonovsky, M., and Komodakis, N. (2018). GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. arXiv.

216. Hoogeboom, E., Satorras, V.G., Vignac, C., and Welling, M. (2022). Equivariant Diffusion for Molecule Generation in 3D. arXiv.

217. Morehead, A., and Cheng, J. (2023). Geometry-Complete Diffusion for 3D Molecule Generation. arXiv.

218. Zhu, J.-Y., and Park, T. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2017 IEEE International Conference on Computer Vision (ICCV).

219. Irwin, J.J., Tang, K.G., Young, J., Dandarchuluun, C., Wong, B.R., Khurelbaatar, M., Moroz, Y.S., Mayfield, J., and Sayle, R.A. (2020). ZINC20-A Free Ultralarge-Scale Chemical Database for Ligand Discovery. J. Chem. Inf. Model. *60*, 6065–6073.

220. Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv.org.

221. Understanding Knowledge Distillation - Ramesh's Blog Available at: https://ramesharvind.github.io/posts/deep-learning/knowledge-distillation/ [Accessed February 23, 2023].

222. Santos-Martins, D., Solis-Vasquez, L., Tillack, A.F., Sanner, M.F., Koch, A., and Forli, S. (2021). Accelerating AutoDock4 with GPUs and Gradient-Based Local Search. J. Chem. Theory Comput. *17*, 1060–1073.

223. O'Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., and Hutchison, G.R. (2011). Open Babel: An open chemical toolbox. J. Cheminform. *3*, 33.

224. O'Boyle, N., and Dalke, A. (2018). DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures.

225. Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A.A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. arXiv.

226. Almahairi, A., Rajeswar, S., Sordoni, A., Bachman, P., and Courville, A.C. (2018). Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data. undefined.

9. Appendix

9.1 List of Abbreviations

Abbreviation	**Explanation**
(V)AE	(Variational) Autoencoder
ACAA1	Acetyl-CoA Acyltransferase 1
AI	Artificial Intelligence
ALL	Acute Lymphocytic Leukaemia
AML	Acute Myeloid Leukaemia
cDNA	Complementary DNA (sometimes also Copy DNA)
CISPA	Center for IT Security, Privacy and Accountability
CLL	Chronic Lymphocytic Leukaemia
CML	Chronic Myeloid Leukaemia
CO_2	Carbon Dioxide
DDIM	Denoising Diffusion Implicit Models
DDPM	Denoising Diffusion Probabilistic Models
DE	Differential Expression

DM	Diffusion Model
DNA	Deoxyribonucleic Acid
DP-CVAE	Differentially Private Conditional Variational Autoencoder
DP-SGD	Differentially Private Stochastic Gradient Descent
eQTL	Expression Quantitative Trait Loci
ESR1	Estrogen Receptor 1
FDA	U.S. Food and Drug Administration
GAN	Generative Adversarial Network
GFC	Group Fold-Change
GRU	Gated Recurrent Unit
KL	Kulback -Leibler (Divergence)
LIMO	Latent Inceptionism on Molecules
LSTM	Long Short-Term Memory
MCF7	Michigan Cancer Foundation-7 (but it references a there established breast cancer cell line)
ML	Machine Learning
MOSES	Molecular Sets
MPP	Methylpiperidinopyrazole
mRNA	Messenger Ribonucleic Acid
MSE	Mean Squared Error
PAINS	Pan-Assay Interfering Compounds
PCA	Principal Component Analysis

QED	Quantitative Estimate of Drug-Likeness
ReLU	Rectified Linear Unit
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
SA	Synthetic Accessibility
SELFIES	Self-Referencing Embedded Strings
SMILES	Simplified Molecular-Input Line-Entry System
SNP	Single Nucleotide Polymorphism
VCAP	Vertebral-Cancer of the Prostate (a cancer cell line)
WGAN	Wasserstein Generative Adversarial Network

9.2 List of Appendix Figures

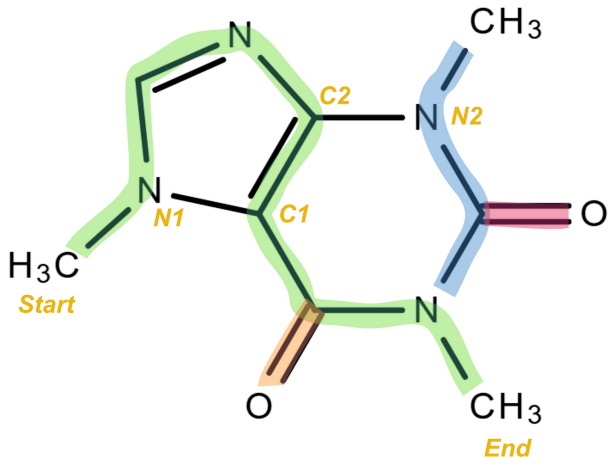
9.3 List of Supplementary Figures

9.4 SMILES, deepSMILES and SELFIES

The Simplified Molecular Input Line System (SMILES) was introduced in 1988 by David Weininger to represent molecules for use with computers. The SMILES notation represents molecules as sequences of characters. Atoms are denoted as their chemical symbols, where in two-letter symbols, the first letter is uppercase and the second letter must always be lowercase. Atoms other than those commonly encountered in organic molecules (C, N, O, S, F, P, B, Br, Cl, I) as well as hydrogen must be enclosed in square brackets, e.g. [Fe]. Charges and attached hydrogens of an atom must be stated inside the square brackets. Molecular bonds are symbolised as "—" for single bonds, "=" for double bonds, "#" for triple bonds and ':' for aromatic bonds. However, single and aromatic bonds are typically not shown. Aromatic atoms are instead symbolised as lowercase versions of their symbol, e.g. *C* is a non-aromatic and *c* an aromatic carbon. Branches are surrounded by parentheses. For branches with further sub-branches, parentheses can be nested. To represent rings, one of their bonds (either a single bond or an aromatic bond) is removed in order to linearise them. Each ring is given a number and this number is appended to the atom opening the ring as well as the atom closing the ring, after the edge has been removed. This helps to recognise the start and end of a ring in its linear form. Disconnected structures as for example ions can be denoted using full stops between the two SMILES of the structures. In the original publication, stereochemistry was deliberately left out to be discussed in the future. Since then, options for stereochemistry have been added to the SMILES notation. *Cis*- and *trans*-configurations around double bonds can be indicated using flanking forward- and

backward-slashes. *Trans*-configurations are indicated by slashes in the same direction, whereas *cis*-configurations are indicated by slashes in opposing directions. To indicate enantiomers, i.e. mirrored versions of a molecule that cannot be superimposed, the symbols @ and @@ were introduced to depict anit-clockwise and clockwise orientation of bound atoms around the chiral centre, respectively. An example of how the SMILES of a molecule can be created is shown in appendix figure 1.

While SMILES are highly flexible, fairly simple and mostly human-readable they do have some shortcomings [163]: SMILES are not unique. Depending on which atom in the molecule is selected to start the traversal, many different SMILES can be generated to describe the same exact molecule. Since there is no consensus on where to start, canonicalisation of this procedure is not unified and different canonicalisation algorithms are not comparable. Additionally, the reliance on paired tokens for opening and closing branches as well as rings make SMILES difficult to work with in generative or reconstructive tasks because small mistakes render them invalid.



Appendix Figure 1: Exemplary generation of a SMILES string. Shown is the SMILES for the molecule caffeine. The highlighted green path annotates the "main path" through the molecule, with start and end indicated. Side branches are coloured differently. Colours in the SMILES match those in the molecule to highlight correspondence.

The latter issue was tackled by the introduction of deepSMILES [224], where paired tokens were removed entirely. Instead, deepSMILES use only closing symbols for branches and rings without corresponding opening symbols. These closing symbols contain information about the length of the preceding branch or ring. To identify the upstream starting atom, a stack is used: whenever a closing symbol is encountered, the corresponding number of atoms

are removed from the stack. The atom that then remains at the top of the stack is the root of the branch or the opening of the ring. However, deepSMILES do not fix the issue of validity. An invalid deepSMILES can be easily generated by adding a closing symbol that removes more atoms from the stack than exist in the molecule, leaving no valid root atom. Additionally, deepSMILES are also not unique and again many deepSMILES exist that describe the same underlying molecule.

To overcome the remaining limitations of deepSMILES, Krenn *et al.* [164] developed a 100% robust string representation called SELF-referencIng Embedded Strings (SELFIES). They guarantee that any SELFIES string is valid, even if the tokens are randomly generated. Additionally, SELFIES are unique, with one string representing a given molecule.

The validity guarantee originates in the fact that the SELFIES tokens all represent rule-vectors, i.e. given a current state of derivation it dictates how the SELFIES token should be replaced, either by an atom or bond and/or the next derivation state. The derivation states connect previous tokens to allowed future actions, making invalid combinations impossible. When randomly mutating a SELFIES token, the resulting molecule is always valid, because essentially an existing construction rule is simply replaced by a different but also valid construction rule.

9.5 Cycle-GAN for drug generation

In the original outline of the DrugDiffusion project, I did not plan to use a diffusion model. Instead, the initial idea was to use a cycle-GAN [218]. CycleGANs were introduced in 2017 to tackle the issue of paired data required to train existing models for the task of image-to-image translation [225]. Image-to-image translation considers an image of a domain X and translates it into an image of domain Y. Often found examples for this task are photographs that are to be converted into paintings of a certain painter's style, images of horses that are turned into zebras, pictures of a summer scenery that is translated into winter, street maps that are created from aerial images or photorealistic scenes that are generated from image segmentation maps. For many of these tasks, paired training data simply does not exist or creating it is very costly and time-consuming. To circumvent supervised training and the requirement for paired data that comes with it, cycle-GANs were proposed. They comprise two GANs, one composed of generator G_{XY} and discriminator D_Y, the other comprising G_{YX} and D_X. The first generator takes as input a sample x from data domain X and

generates a sample $G_{XY}(x) = \widehat{y} \in \widehat{Y}$ were $\widehat{Y}$ is indistinguishable from the data distribution of domain Y. Inversely, the second generator receives a sample $y \in Y$ and generates a sample $G_{YX}(y) = \widehat{x} \in \widehat{X}$, with $\widehat{X}$ following the same distribution as X. To ensure the realism of the samples produced by the generators, the discriminators are trained jointly with the generators: D_X and D_Y are trained to classify between real and fake instances of domain X and Y, respectively. However, there still remain infinitely many possible mappings between instances of X and instances of Y, the majority of which is not meaningful. In addition to that, GANs are known to be prone to mode collapse, in which case all inputs of one domain could be mapped onto the same output of the other domain, if that particular output fools the discriminator well. Thus, the authors added another additional constraint to the training procedure. It is inspired by language translation where, when translating a sentence from one language to the other and then back again, we expect to arrive back at the original sentence. The constrained is called the *cycle consistency loss* and it enforces $G_{YX}(G_{XY}(x)) \approx x$ and $G_{XY}(G_{YX}(y)) \approx y$.

The reason why cycle-GANs were appealing in the context of property-based molecule generation is that for some properties, not much paired data is available. While large molecule databases are annotated with more simple physico-chemical properties, data is still lacking for complex biochemical or systems-biological properties such as protein binding-affinities or molecule-induced gene-expression. However, in particular for the gene-expression case, a lot of unpaired data is available. The difference to the image-to-image translation task is, that although the sets between which the translation happens are referred to as different domains, they are still both images. That inherently provides a structure to the generator which to maintain, e.g. the scenery, which gives a lot of guidance on what to modify and what to keep. When translating from gene-expression to a molecule, however, such inherent structure is not given, since they come from distinctly different domains. Thus, the discriminator loss may guarantee, that the samples generated follow the same distribution as the target domain and therefore appear realistic and the cycle consistency loss may guarantee, that the generation does not collapse onto the same output, but they cannot vouch for a meaningful mapping between the gene-expression and the molecule that induced it and vice versa. In such a scenario a third loss would be required, one that provides contextual guidance much like the general scenery does in the image case. The original outline envisioned a third loss component provided by a function f, where f is a

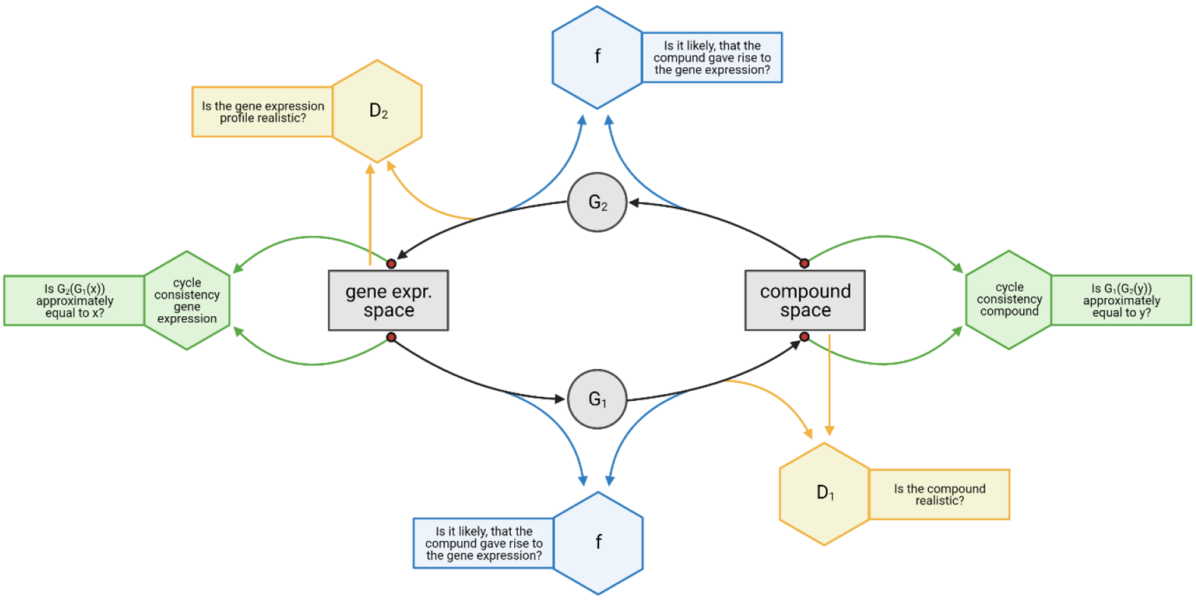
deep neural network that scores how likely a given drug induced a provided gene-expression profile. Such a third loss component would add contextual meaning to the mapping. A loss like this was used by Méndez-Lucio *et al.* for their conditional GAN which is the same property predictor that I ended up using for guidance in experiment 5. The originally envisioned drug cycle-GAN is illustrated in appendix figure 2. The reason why we eventually decided against using a cycle-GAN is its one-to-one mapping: The cycle consistency loss enforces, that any instance $x \in X$ is mapped only to a single instance $y \in Y$ and vice versa. While that has the beneficial mentioned effect of counteracting mode collapse, it also restrains the generative process such that the generator only generates a single output for any given input. In the context of image-to-image translation that is often not a problem, but for the generation of molecules this is highly restrictive. While the proposed molecule may exhibit the desired target property, e.g. induction of a given gene-expression profile, it may fail early scans for toxicity or might not even be synthesisable. Therefore, generating a larger variety of candidates is essential to increase the odds of a few passing the required screening tests.

The insufficient applicability of cycle-GANs to problems that require many-to-many mappings was also noticed by Almahairi *et al.* [226], who proposed *augmented* cycle-GANs. They suggest not to learn a mapping from one instance $x \in X$ to a single other instance $y \in Y$, but instead to learn a mapping between tuples. More precisely, a mapping between $(x, z_y) \in X \times Z_y$ and $(y, z_x) \in Y \times Z_x$, where Z_y and Z_x are both latent variables sampled from a Gaussian. The latents are auxiliary variables that encode variable aspects of the generated target sample. An illustrative example provided in the paper is the task of changing a male face x into a female face y. In this case, y is a female face that resembles the original face, while the auxiliary latent variable z_y represents interchangeable features such as hair style and length. Inversely, z_x contains all features of the male face, e.g. beard style and length, that are lost when transitioning to y, but that are required to reconstruct x. With this approach, an instance $x \in X$ can be mapped to many instances $y \in Y$ by sampling z_y (one-to-many mapping), while an instance $y \in Y$ can be mapped to multiple $x \in X$ by sampling z_x, allowing overall a many-to-many mapping.

The augmented cycle-GAN relies on several neural network components. A generator G_{XY}, such that $\widehat{y} = G_{XY}(x, z_y)$ and an encoder E_X which computes $\widehat{z_x} = E_X(x, \widehat{y})$ and for the

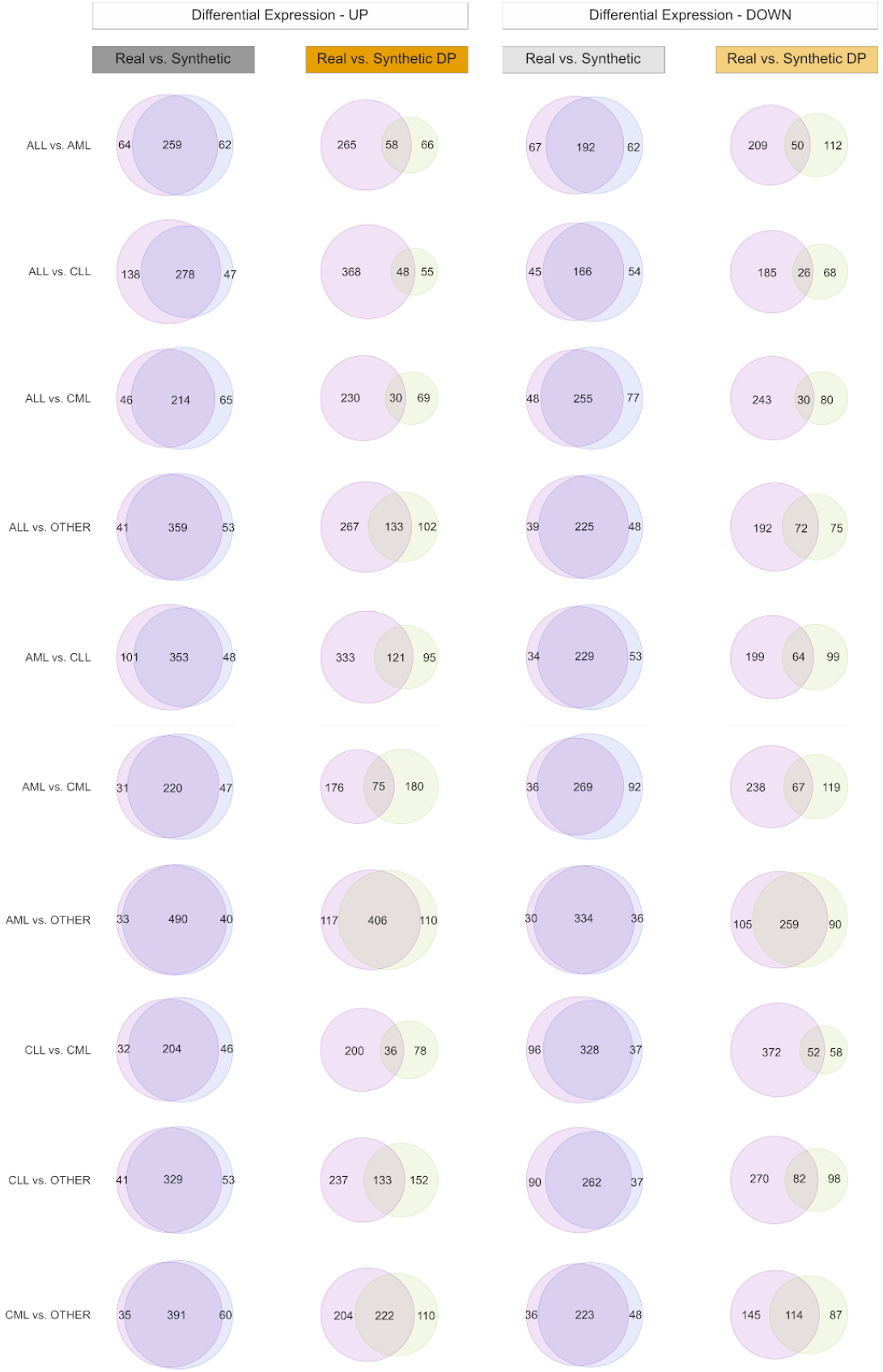
inverse direction a generator G_{YX} that generates $\widehat{x} = G_{YX}(y, z_x)$ and a corresponding encoder E_Y which computes $\widehat{z_y} = E_Y(y, \widehat{x})$. Additionally, four discriminators D_X and D_Y, as used in the cycle-GAN approach, as well as D_{z_x} and D_{z_Y} that are used to regularise the generated latents to follow the Gaussian prior. Given that samples are generated as pairs, the cycle-consistency loss in augmented cycle-GANs is realised as two separate losses: The first is the cycle-consistency on x or y with $L_{CYC}^{X}(G_{XY}, G_{YX}, E_X) = \mathbb{E}_{x\sim p_d(x),\, z_y\sim p(z_y)}||x' - x||_1$, where $x' = G_{YX}(\widehat{y}, \widehat{z_x})$, $\widehat{z_x} = E_X(x, \widehat{y})$ and $\widehat{y} = G_{XY}(x, z_y)$ and analogously when starting from y. The second is a cycle-consistency restraint on the auxiliary latent variable, with $L_{CYC}^{z_y}(G_{XY}, E_Y) = \mathbb{E}_{x\sim p_d(x),\, z_y\sim p(z_y)}\left|\left|z_y' - z_y\right|\right|_1$, where $z_y' = E_Y(x, \widehat{y})$ and $\widehat{y} = G_{XY}(x, z_y)$ and analogously for the reconstruction of z_x.

Translating this back into the molecule context, augmented cycle-GANs would enable the generation of multiple molecules based on an input property. And yet, we decided against using them for two reasons. Firstly, it is not intuitively apparent how to expand the cycle-concept to incorporate multiple properties into the molecule generation. Given the complexity of a drug's possible effects in and interactions with the organism, multi property-based generation is indispensable to reach reasonable utility of the tool. The second reason was the demonstrated performance. While the examples given in the paper demonstrated that the augmented cycle-GAN was indeed capable of generating a variety of different outputs given the same input, the image quality was not outstandingly good and artefacts were very common. Latent diffusion models on the other side have demonstrated their extraordinary capabilities in generating images. This, together with the more intuitive incorporation of multiple properties led us to eventually choose latent diffusion models rather than augmented cycle-GANs.

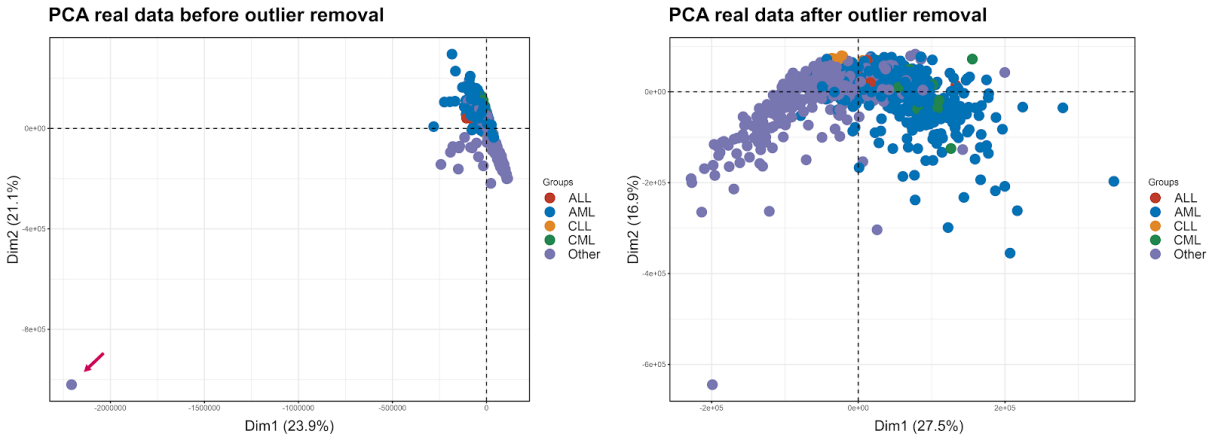


Appendix Figure 2: Initially proposed cycle-GAN. The core comprises the two generators G_1 and G_2 (grey) that map data points from the gene expression space to the compound space and vice versa. Several additional components are meant to ensure a meaningful mapping: The discriminators D_1 and D_2 (yellow) assess whether the generated sample looks like a realistic molecule or gene expression profile, respectively. The two cycle-consistency losses (green) ensure that samples are mapped back to themselves after traversing both generators. A function f (blue), symbolising a trained neural network, evaluates how likely a given compound induces a given gene-expression profile and therefore ensures a contextually meaningful mapping between the domains.

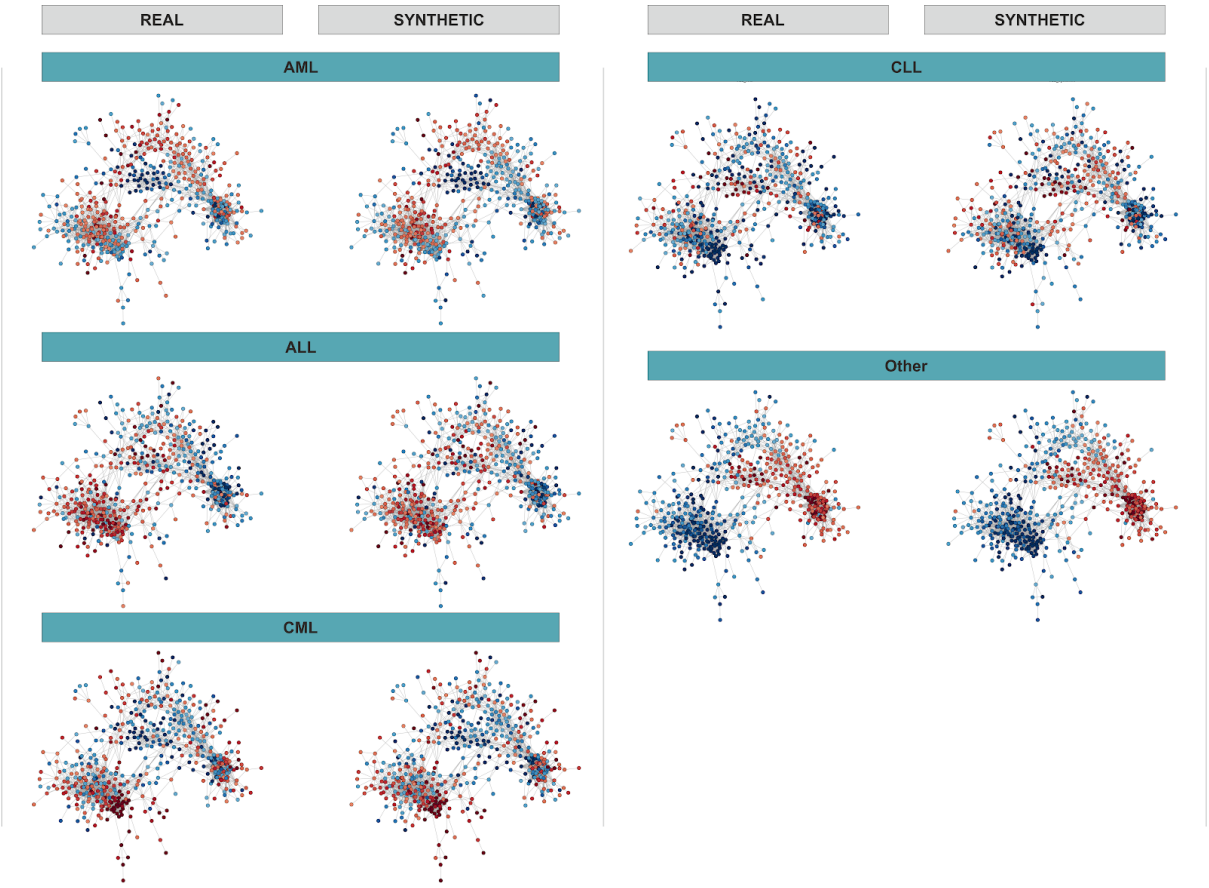
9.6 Supplementary Figures



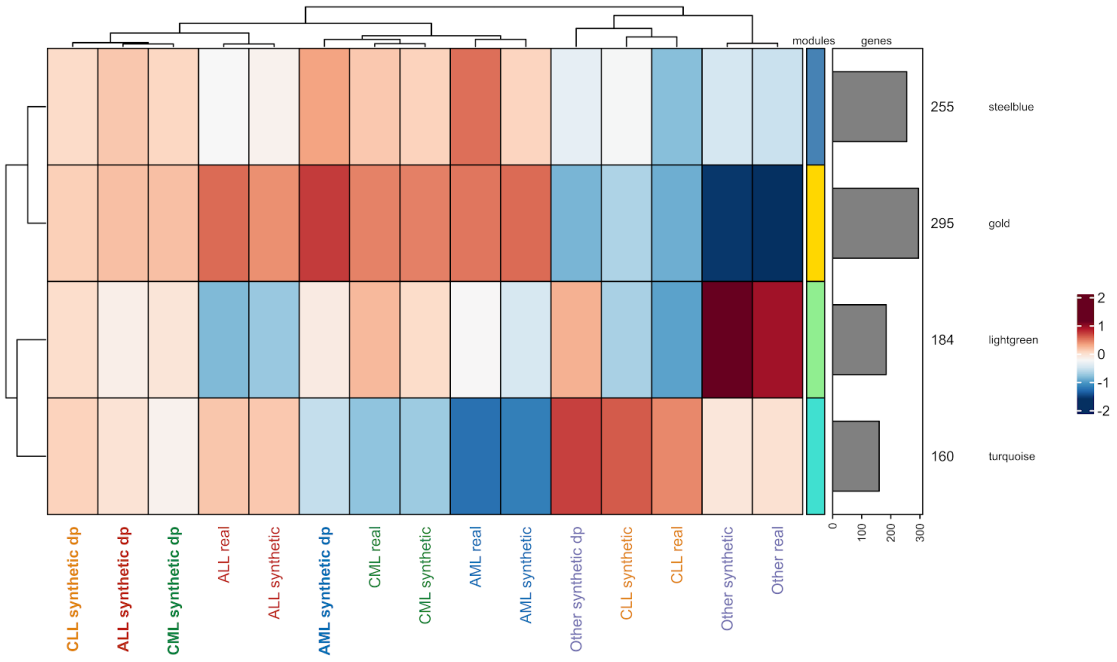
Supplementary Figure 1: Venn diagrams of up- and down-regulated DE-genes across all condition comparisons. Overlap between DE-genes in the real data (pink) and the non-private synthetic data (blue) as well as the private synthetic data (light green) are shown for each condition pair (rows). The two leftmost columns show up-regulated DE-genes for the comparisons real vs. non-private synthetic and real vs. private synthetic (DP), respectively. Analogously, the two rightmost columns show the down-regulated DE-genes. Numbers indicate the number of DE-genes, intersection areas show DE-genes that are shared between the datasets. DP: differential privacy; AML: acute myeloid leukaemia, ALL: acute lymphocytic leukaemia, CML: chronic myeloid leukaemia, CLL: chronic lymphocytic leukaemia. "Other" comprises a variety of different conditions (no leukaemia) and controls.



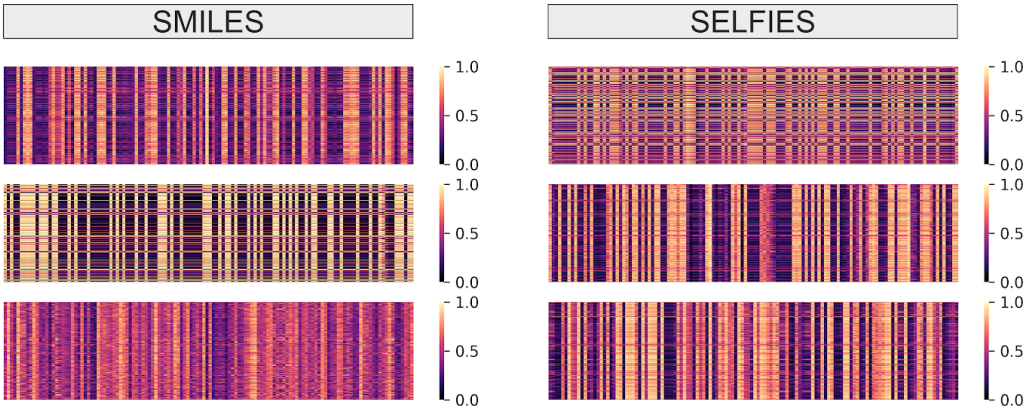
Supplementary Figure 2: PCA of real data before and after outlier removal. The removed sample is highlighted in the left PCA with a red arrow. The sample originated from the category "other".



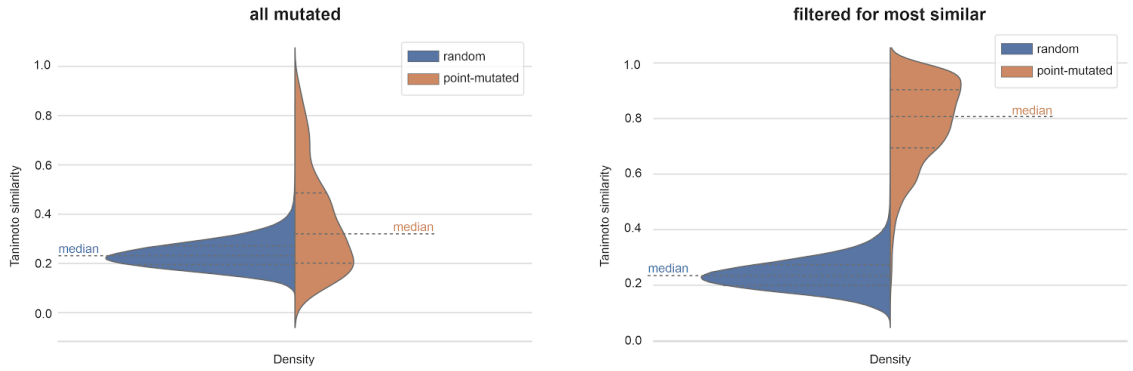
Supplementary Figure 3: Integrated network coloured by GFC. The integrated co-expression network is shown for all conditions and for both real and non-private synthetic data. Each node represents a gene and is coloured by its GFC in the respective dataset and under the given condition. GFCs range from >= 2 (dark red) to <= − 2 (dark blue) with the midpoint being 0 (white). AML: acute myeloid leukaemia, ALL: acute lymphocytic leukaemia, CML: chronic myeloid leukaemia, CLL: chronic lymphocytic leukaemia. "Other" comprises a variety of different conditions (no leukaemia) and controls.



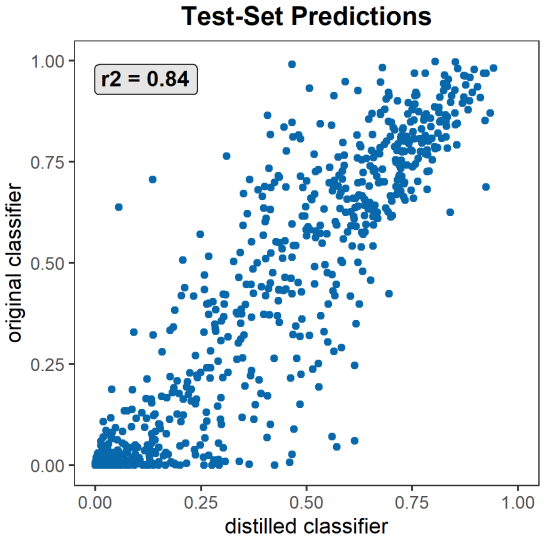
Supplementary Figure 4: Cluster heatmap for real, non-private and private synthetic data. Heatmap shows the mean GFC for each gene-cluster across conditions. Conditions are clustered based on their mean GFC patterns using hierarchical clustering. Barplots on the right indicate the number of genes present in each cluster. Private synthetic data is highlighted in bold. dp: differential privacy; AML: acute myeloid leukaemia, ALL: acute lymphocytic leukaemia, CML: chronic myeloid leukaemia, CLL: chronic lymphocytic leukaemia. "Other" comprises a variety of different conditions (no leukaemia) and controls.



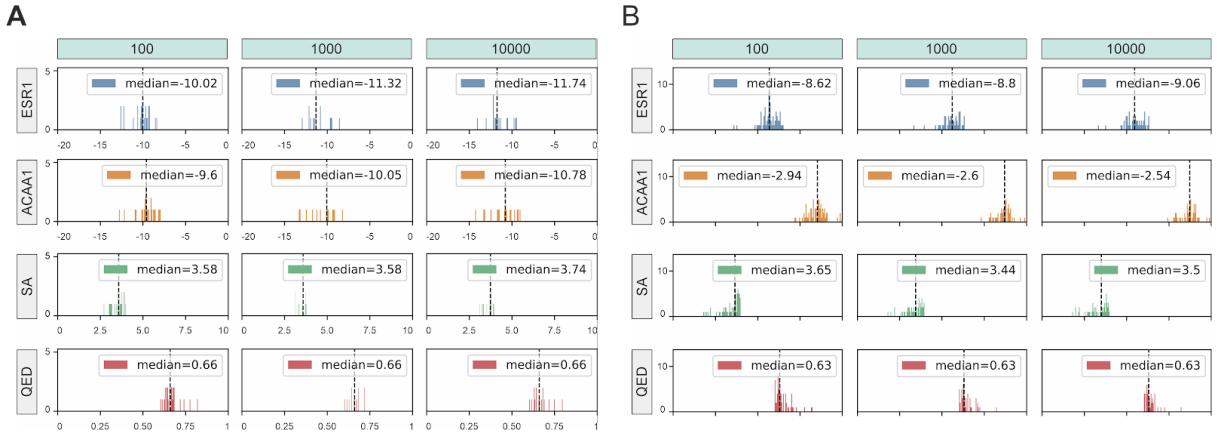
Supplementary Figure 5: Latent space utilisation of LSTMs trained on full MOSES with latent size 128. For each string input type, three latent space utilisation plots are shown, representing the three different seeds used for training. In each heatmap, rows are molecules and columns are latent dimensions. All encoding values were scaled using min-max scaling within each heatmap to make patterns comparable.



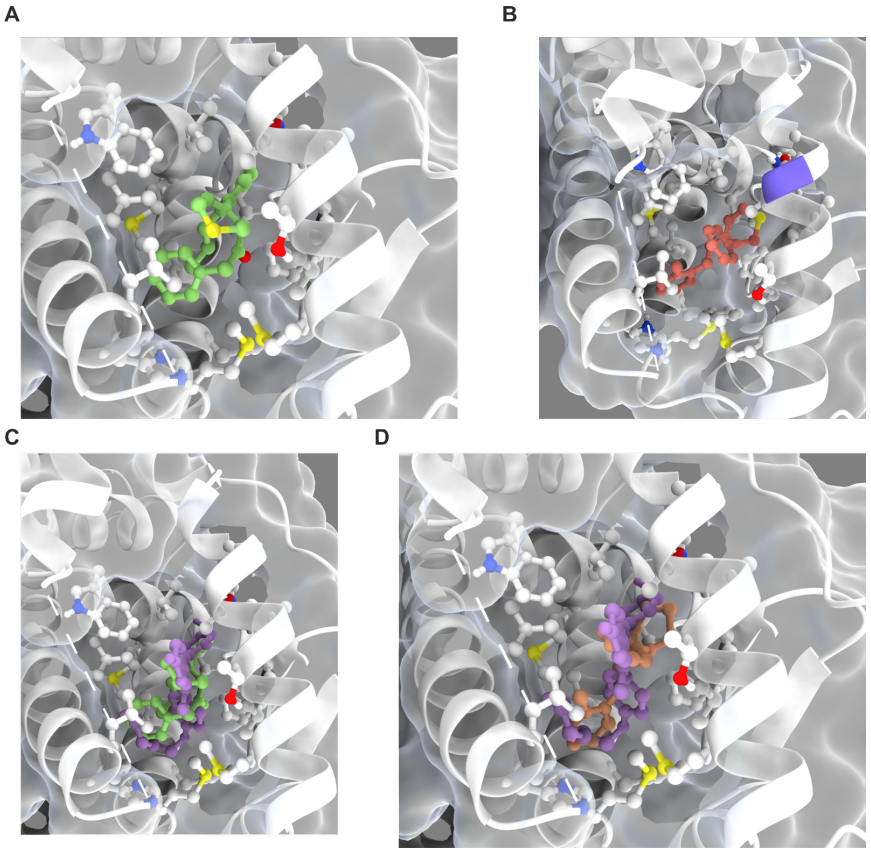
Supplementary Figure 6: Tanimoto similarity of SELFIES after point mutation. Left: Tanimoto similarities of SELFIES after introducing a single random point mutation to all SELFIES and comparing each to the original version (orange) and the Tanimoto similarities of random molecules (blue). Right: The blue distribution is the same as before, while the orange distribution results from introducing a single point mutation to each SELFIES 20 times and selecting only the 5 most similar ones to the original molecule. For both experiments, 1000 molecules were randomly selected from the test set.



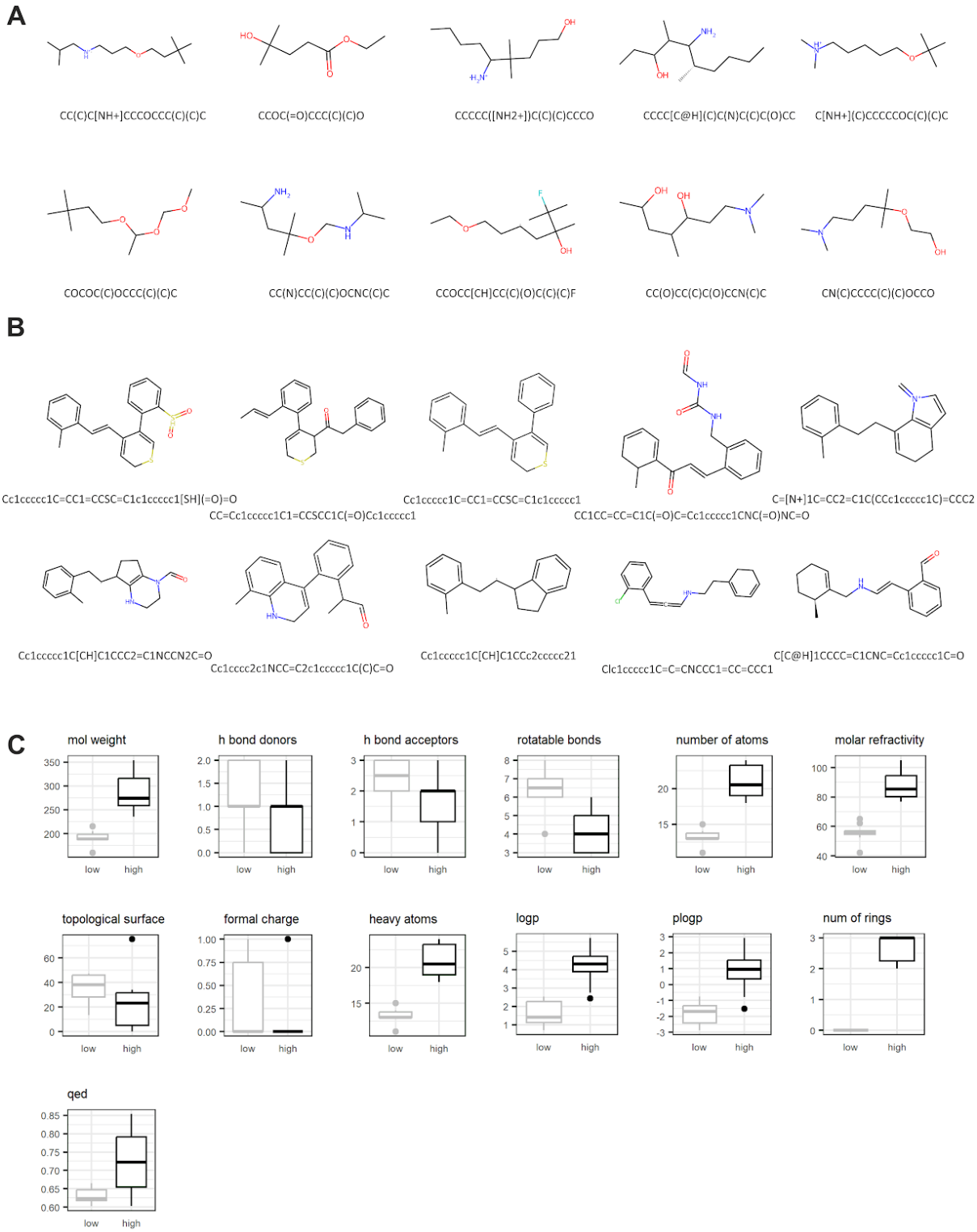
Supplementary Figure 7: Test set prediction of the original vs. the distilled predictor. Predictions were made on a test set of randomly sampled gene-expressions and molecules from the VAE latent space previously unseen by the distilled predictor.



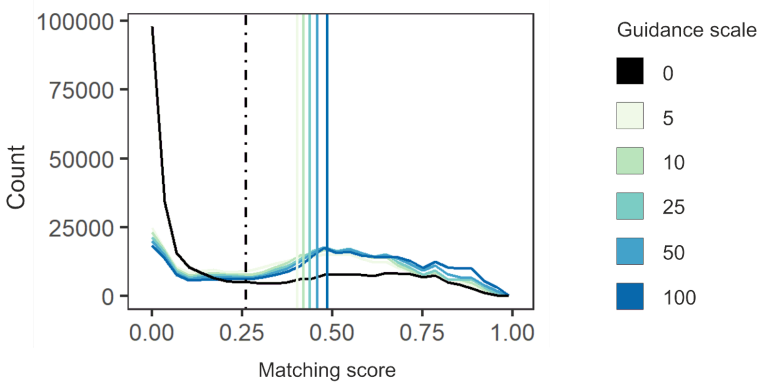
Supplementary Figure 8: Sampling LIMO with 100, 1000, 10000 steps. Shown are predicted *ESR1*- and *ACAA1*-scores as well as computed synthetic accessibility (SA) and drug-likeness (QED) of molecules generated with LIMO using different number of optimisation steps as indicated on top. All generated molecules were filtered for good synthesisability (SA < 4) and high drug-likeness (QED > 0.6). A) Results of scenario 1, where generated molecules were meant to show high affinity (represented by a low score in the plots) to both target proteins; B) results of scenario 2, where generated molecules were meant to have high affinity to *ESR1* (represented by low predicted score) and low affinity to *ACAA1* (represented by high predicted score).



Supplementary Figure 9: Visualisation of second and third best generated ligand in *ESR1*-binding pocket. **A)** and **B)** show the second and third best generated ligand (based on simulated docking), respectively, in the binding pocket alone, while **C)** and **D)** show them together with the known ligand MPP. Visualisations were done with Chimera X and poses were predicted with Autodock-GPU.



Supplementary Figure 10: Structural differences of molecules generated for low and high *ESR1*-affinity. The model generated 10,000 molecules for low and for high *ESR1*-affinity. They were filtered for SA < 4, QED > 0.6 and no cycles with less than 5 or more than 6 atoms and the 10 top ranked hits for each scenario, i.e. low (**A**) and high (**B**) binding affinity, were selected. Several molecular properties were calculated for the two sets of molecules (**C**). h bond: hydrogen bond; plogp: penalised-logP; qed: Quantitative Estimate of Drug-Likeness.



Supplementary Figure 11: Gene-expression guidance including molecules sampled without guidance. Predicted matching scores of molecules generated for given gene-expression profiles. Colours indicate different guidance scales used during generation, vertical lines indicate the mean predicted scores (the higher the better). The black line represents the predicted scores of molecules generated without guidance, i.e. random molecules and the black dashed line is their mean.