Bachelor's Thesis Computing Science

Exploring the Chemical Universe with Spiking Neural Networks

Opportunities and Limitations in the Context of Drug Discovery

Philipp Küppers
s1073738

June 27, 2024

First supervisor/assessor:
Dr. Twan van Laarhoven

Second assessor:
Dr. Francesca Grisoni

Radboud University

Abstract

When identifying promising molecules to use in drugs, Artificial Neural Networks (ANNs) only explore a fraction of the chemical universe due to their high computational costs, despite advancements in hardware and model architectures. Inspired by the workings of our brain, Spiking Neural Networks (SNNs) offer low computational complexity and energy consumption when run on neuromorphic hardware. This hardware bypasses the limitations of the van-Neumann architecture used for ANNs caused by separating memory and a processing unit. In this thesis, we evaluate SNNs for their potential to discover more of the chemical universe through two commonly used methods in drug discovery: virtual screening and de novo design. Thus, we first implement a three layered Spiking Neural Network on the neuromorphic Intel Loihi 2 processor for predicting the bioactivity of molecules (virtual screening). Subsequently, we look at the ability of a generative SNN called SpikeGPT to learn the chemical language by generating valid, unique and novel molecules from scratch (de novo design). The conducted research shows that SNNs implemented on neuromorphic hardware can perform 2.7 to 4.1 times more predictions using more than 100 times less energy than ANNs, while matching their accuracy when predicting bioactivity. SpikeGPT manages to match the state of the art in learning the chemical language, with a computational complexity that needs to be experimentally verified in future research. The results suggest that Spiking Neural Networks could aid in considering unprecedented numbers of molecules while significantly reducing the energy footprint of deep learning methods used in drug discovery.

Contents

Chapter 1

Introduction

When developing a new drug, pharmaceutical scientists try to find a combination of molecules with the right properties in a search space of up to 10^{60} small molecules [1]. This space, consisting of all possible molecules that could be theoretically synthesized in a lab, is often referred to as chemical space or universe. The largest databases of known molecules are in the size of hundreds of millions [2, 3], meaning that most of the chemical space is still uncharted territory. Besides the overwhelming search space, it is nearly impossible to evaluate the properties of molecules without costly lab experiments [4]. These issues make drug discovery an inefficient process, which has become exponentially more expensive and the costs have doubled within the past 10 years [5, 6]. To put this into context, the average price for bringing a new drug onto the market between 2014 and 2018 was estimated at $1335.9 million [7], with an overall success rate of clinical drug development of 10%-15% [4].

A recent effort to mitigate the issues in drug discovery is the use of Artificial Neural Networks (ANN) for the efficient exploration of the search space. These networks do so by predicting potentially suitable compounds for a certain drug (called hit candidates) before pharmaceutical scientists proceed to investigate their properties via synthesis in a lab [8]. However, ANNs are only able to consider molecules in the range of billions when finding hit candidates [9] due to their high requirements of compute resources, energy and time [10, 11]. For example, in natural language processing, it is estimated that the large language model ChatGPT consumes half a million kilowatt-hours of electricity per day for inference [12] - more than 17,000 average homes in the U.S. These high computational costs are largely due to the limits of the van-Neumann architecture that ANNs are run on. Due to the separation of processing unit and memory, the communication of the two creates a bottleneck (called the van-Neumann bottleneck [13]) that is getting

harder and harder to overcome for hardware designers. Another issue are the thermal limitations for silicon chips that cause chip manufacturers to reduce clock speeds and thus performance [14]. These issues are already noticeable by a slowing down in the projections of Moore's law since 2010 [15].

Spiking Neural Networks (SNN), often praised as the third generation of neural networks [16], could address the issues ANNs face. By mimicking the functionality of our brain much closer than ANNs, they are shown to be much faster and energy efficient when implemented on special neuromorphic hardware that overcomes the van-Neumann bottleneck by doing in memory computations [17, 18]. Despite many reported problems with training SNNs and performance degradation on certain tasks, they have shown great promise for robotics control, audio processing and autonomous cars [17, 18]. For instance, SpikeGPT achieves comparable performance in natural language processing tasks with at least 32x fewer operations [19].

Besides efficiency, there are other arguments that make SNNs a promising candidate for us to try for drug discovery. SNNs can be formulated as Recurrent Neural Networks (RNN) [20], which are still the status quo neural network type for AI aided drug discovery [8, 21], making us optimistic to see a comparable performance by SNNs. Furthermore, SNNs can directly use binary molecular fingerprints as input, which are commonly used for virtual screening [21]. Last but not least, ANNs leave room for improvement in the quality of discovered compounds [8, 21]. For example, in virtual screening, machine learning models still struggle with delivering high and consistent accuracy in predicting properties of molecules that are not synthesized in the lab yet. Here, the architecture of SNNs could potentially result in better results than the state of the art.

This thesis tries to address the outlined issues of ANNs for drug discovery by answering the following research question: Can Spiking Neural Networks (SNN) consider more of the chemical universe than ANNs when identifying hit candidates through a lower energy consumption and faster inference times while matching their performance? More specifically, we use two SNNs for two techniques of finding hit candidates called virtual screening and de novo design, which are the most used computational methods in drug discovery [8, 22–24]. In *Virtual Screening* [25], a neural network is trained on a database of known molecules to *predict* desired properties of unknown molecules. From these predictions, a list of hit candidates is created, which are used for lab experiments. The second technique is *de novo design* [8], where the model is also trained on a database of known molecules but is instead used to directly *generate* new (novo) molecules, which might have never been synthesized. To answer the research question, our idea consists of a Spiking Neural Network that predicts bioactivity and runs on the neuromorphic Intel Loihi 2 processor for virtual screening. For de novo design, we use the existing SpikeGPT

architecture [19].

In summary, this thesis shows that a three-layered SNN can match the accuracy of a comparable ANN for predicting bioactivity, while performing 2.7 to 4.1 times more predictions and using more than 100 times less energy. For de novo design, SpikeGPT matches the state of the art in generating valid, unique and novel molecules. Thus, this thesis makes the following contributions:

- First ever evaluation of Spiking Neural Networks for de novo molecule design (Chapter 4.2).
- First ever implementation of a Spiking Neural Network on neuromorphic hardware for molecular property prediction, to experimentally measure energy consumption and inference latency (Chapter 4.1).
- Analysing the impact of different temporal dimensions when using a rate encoding in molecular property prediction (Chapter 4.1).

Chapter 2 explains the required knowledge to understand this work, followed by the methodology (Chapter 3) used to achieve the presented results (Chapter 4). Finally, the results are discussed in detail (Chapter 5) and compared to related work (Chapter 6. Last but not least, we summarize our results (Chapter 7) and give directions for future research are outlined (Chapter 8).

Chapter 2

Preliminaries

In this chapter we will outline the required knowledge for the reader to understand this research.

2.1 Drug Discovery

As outlined in Singh *et al.* [26], the drug discovery process consists of 5 stages and takes on average 12-15 years. First, researchers try to understand the disease and identify possible proteins or other structures the drug should target in order to fight the disease. Second, scientists search for molecules (sometimes referred to as 'hit candidates'), which they believe could be used in the drug to evoke a satisfying effect to a pharmacologically-relevant target. This is more and more done using computational methods and also the stage which we place our work in. In the third stage, the drug candidates are first evaluated in a laboratory dish or test tube (in vitro) and then on living organisms like animals (in vivo). Fourth, there is a clinical stage that evaluates the drugs in humans. And finally, the drug is reviewed, approved and marketed.

2.2 Molecule Representation

In order to identify hit candidates using neural networks, different digital representations of molecules are used [8]. The most commonly used representations, which we also choose in this research, is to represent molecules as Simplified Molecular Input Line Entry Systems (SMILES) strings [27]. SMILES are strings of characters, following a specific syntax. They are

Figure 2.1: Process of converting a molecular graph into a SMILES string.

constructed from the molecular graph of a molecule (see also Figure 2.1). This molecular graph consists of vertices corresponding to the atoms of the molecule and edges corresponding to the chemical bonds connecting the atoms. A string is constructed from this molecular graph by traversing the graph with a search algorithm like depth first search and converting each encountered vertex and edge into the corresponding characters of the SMILES string syntax.

It is important to note that depending on the algorithm (and hence traversal order) used to traverse the molecular graph, one molecular graph can result in different SMILES strings. A common way to count the number of unique SMILES strings for a molecule, is by using a standard traversal of molecular graphs for each smile. This process is referred to as canonicalisation. The advantage of SMILES strings is their compact one dimensional string format, allowing for processing and storing billions of molecules. Nevertheless, SMILES strings do not retain 3D coordinates for individual atoms, allow for chemically invalid molecules and do not directly encode any chemical information.

2.3 Molecular Fingerprints

In contrast to SMILES strings that describe the entire molecular graph of a molecule, molecular fingerprints directly encode chemical substructures and properties relevant for drug discovery. This fingerprint is a bit vector, indicating the presence or absence of a certain number of molecular substructures or properties in the molecule. Thus, fingerprints do not capture how often a substructure is present, making it possible that multiple molecules result in the same fingerprint. They are primarily used for predicting properties of molecules instead of generating molecules from scratch. Even though there are many different fingerprints [28], we choose the commonly used Morgan [29] and MACCS keys fingerprints [30]. Morgan fingerprints include a variable number of chemical structures, with a radius determining the maximum size of each substructure. In MACCS keys, each bit corresponds to a certain pre-defined molecular substructure or property, defined by MDL

Information Systems.

2.4 Virtual screening

Virtual screening is about screening a large database of molecules to select a number of hit candidates based on certain properties. Machine learning can be used to predict properties, like the toxitivity of molecules, which would need to be experimentally tested in a laboratory. In this thesis, we focus on predicting bioactivity, which quantifies how well molecules would irreversibly bind to the targeted receptor (a receiving protein molecule). The state-of-the-art in the field consists of Support Vector Machines, Decision Trees and Artificial Neural Networks [21, 31].

2.5 De novo design

Another way of identifying hit candidates using machine learning is de novo design [8]. A neural network is trained on a large number of molecules represented as strings (often SMILES strings) and tasked to generate hit candidates from scratch (de novo). One can think of this as learning the 'chemical language' using machine learning algorithms. The syntax of this language are the chemical rules that specify 'chemically valid' molecules. The semantics are the different high-level properties of a molecule like toxicity or molecular weight. De novo design is then about generating molecules with the correct syntax, that are equipped with the right semantics. The state-of-the-art is made up of Long Short-Term Memory (LSTM) network, transformers, variational autencoders and generative adversarial networks (GANs) [32, 33]. Looking at the partially sequential structure of SMILES strings, one would expect transformers [34] to perform significantly better than LSTMs [35] due to the inductive bias of LSTMs for structural data. However, LSTMs and Transformers are showing comparable results in drug discovery when using SMILES strings [36, 37].

2.6 Spiking Neural Networks

Spiking Neural Networks (SNNs) [17] are part of the field of neuromorphic computing, which refers to all computational methods that are closely brain inspired. The term was first established by Carver Mead in 1990, who created the first analog silicon retina and cochlea [38]. The processing advantages our brain offers, are more and more translated in recent times to computational

methods in other tasks such as audio processing and autonomous driving [17]. Our brain offers exceptional computational power at ultra low energy in comparison to computers with a van-Neumann architecture. An example is a comparison between the world's most powerful supercomputer in June 2022, the USA's Frontier. It operates at the same range of about 1 exaFlops as our human brain drawing 21 megawatts of power, while our brain uses at most 20 watts for the same computational power [39]. This makes humans a million times more power efficient than the most powerful computer we can build. Despite all of these advantages, the field of SNNs and neuromorphic computing has mainly remained under the shadow of Artificial Neural Networks run on computers with the van-Neumann architecture since its establishment in 1990. Despite increasing interest in the field, magnitudes less (financial) resources haven been put into research, which can be seen in most of the research being in an early proof-of-concept stage or not publicly available [17].

2.6.1 Information encoding

In Spiking Neural Networks information is represented much closer to how our brain works than regular ANNs. While in our brain information is communicated via electrical impulses of approximately 100 mV in amplitude [40], SNNs simplify this behaviour by representing a voltage burst as 1 (called spike) and its absence as 0, which is far simpler to represent in hardware than a floating point value used in ANNs. This is especially advantageous as neurons spend most of their time without emitting a voltage burst, only emitting 0's. The resulting sparse data structures are cheap to store and can be efficiently used in computations, as most values are multiplied by 0. Together with neuromorphic hardware that is specifically designed for modelling different neuron types, as well as working with binary spikes, computations are done extremely efficient. These spikes are used in various binary encoding schemes, which draw inspiration from how our brain encodes information using voltage bursts [41]. The sequences of 1s and 0s that are flowing through the network are often referred to as spike trains.

For our spiking neural networks in virtual screening, we want to give a bit more background about the rate encoding used, which is biologically inspired from how our brain processes visual input. Thus, information is encoded as the average number of spikes over a given time interval T. To illustrate, we provide an example of rate encoding in a camera, capturing the environment. Each frame consists of 784 binary pixels, representing black or white, which are mapped to one input neuron each. As the camera moves around, the pixel values change over time. Information is now encoded as the average number of spikes over the last 2s.

2.6.2 Leaky Integrate And Fire Neuron

For all our SNNs we use the Leaky Integrate and Fire (LIF) neuron model. The neuron has an internal value, modelling the biological membrane potential. This membrane potential in the brain refers to the electric potential inside a cell with respect to the outside. One can think of this membrane potential as memory (or state), similar to Recurrent Neural Networks (RNN) cells. This similarity allows for describing SNNs as RNNs [20]. The state of a spiking neuron at a time step t is the membrane potential $u[t]$. After each sample that is passed to the network, we reset the membrane potential. When a neuron receives Spikes, it increases its membrane potential, until it reaches a certain threshold, resulting in the neuron firing. After firing, the membrane potential is reset to a voltage, depending on the neuron implementation. The reason the neuron is called "leaky" is that the membrane potential decays at a certain rate, in the absence of spikes. There are small differences in how libraries implement the functionality of the neuron. In this research we use the LIF neurons implemented in the lava-dl libary and the spikingjelly library. Lava-dl is the only publicly available library that allows for directly running SNNs on neuromorphic hardware [42] and spikingjelly is the library used in SpikeGPT, we use for de novo design.

Current Based LIF neuron

The current based LIF neuron in lava is modeled using four equations.

$$u[t] = (1 - \alpha_u)\, u[t-1] + x[t] \tag{2.1}$$
$$v[t] = (1 - \alpha_v)\, v[t-1] + u[t] + \text{bias} \tag{2.2}$$
$$s[t] = v[t] \geq \vartheta \tag{2.3}$$
$$w[t] = v[t]\,(1 - s[t]) \tag{2.4}$$

The membrane potential u at time t, is influenced by the input current $x[t]$ and decays by α_u (see equation 2.1). In contrast to a simple LIF neuron, the synaptic integration potential v at time t is used to determine when the neuron fires and represents the accumulation of incoming synaptic currents within the neuron (see equation 2.2). If the Synaptic integration potential exceeds the threshold ϑ, the neuron emits a 1, indicating that the neuron should fire (see equation 2.3). After the neuron fires, the synaptic integration potential $v[t]$ is set to $w[t]$ (see equation 2.4).

Spiking Jelly LIF neuron

In SpikeGPT, the spiking jelly LIF neuron implementation is used, which models the neuron with a neuronal charge, fire and reset equation.

$$u[t] = v[t] + \alpha_u \cdot (Y[t] - H[t-1] - U_{reset}) \tag{2.5}$$
$$s[t] = \mathcal{O}(u[t] - \vartheta) \tag{2.6}$$
$$v[t] = u[t] \cdot (1 - s[t]) \tag{2.7}$$

For the neuronal charge (see equation 2.5), the membrane potential u of the neuron at time t decays by α_u. Y is the input to the neuron (more details of the Y can be found in [19]) and U_{reset} is set to 0 as done in [19]. The neuron firing is modelled in equation 2.6. The neuron fires at time t, if the membrane potential u crosses the firing threshold ϑ, which is set to 1 done in [19]. This firing behaviour is modelled in the forward pass of the backpropagation algorithm with a heavyside function defined by

$$\mathcal{O}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.8}$$

During the backward pass this function is replaced by the arctangent surrogate function function [43] with gradient g' and primitive g defined as

$$g'(x) = \frac{\alpha}{2(1 + (\frac{\pi}{2}\alpha x)^2)} \tag{2.9}$$
$$g(x) = \frac{1}{\pi}\arctan(\frac{\pi}{2}\alpha x) + \frac{1}{2} \tag{2.10}$$

to overcome the non-differentiability of the heavyside step function during backpropagation. The values of α is set to 2. After firing, the membrane potential u is reset to v (see equation 2.7).

2.6.3 Training Spiking Neural Networks

In ANNs, networks are trained using the de-facto standard backpropagation algorithm [44]. However, due to the non-differentiability of the heavyside function used to model the spiking behaviour of the LIF neuron, the standard backpropagation algorithm is not compatible with SNNs without modifying them. This is also the case for all other neuron types used in the field of spiking neural networks [17, 18]. Researchers have come up with more and more different methods to mitigate this issue. Due to the lack of attention and

funding of the field in comparison to ANNs, training algorithms are known to lack behind normal backpropagation algorithms by being slower and much harder to implement [17, 18]. At first sight, a non-ideal attribute for the goal of this thesis. However, as most papers do not report concrete measurements for the training speed, it was one of our priorities to measure the training speed. A quite popular training solution is based on converting ANNs to SNNs [17] or distilling knowledge from ANNs to SNNs [45], which enables much faster training times. However, as this means that the performance is theoretically bound by the ANN, we choose against this approach.

For molecular property prediction, we train our SNNs using the default training algorithms to directly train SNNs: Spike Layer Error Reassignment (SLAYER) 2.0. This method is an enhanced version of the SLAYER training algorithm [46].

SpikeGPT is trained using backpropagation. This is made possible through using surrogate gradients to estimate the gradients, for parts of the functions used in the SNN which are undefined for certain parts.

2.7 SpikeGPT

The SpikeGPT model is based on the RWKV language model [47] and is the largest backpropagation-trained SNN model to date. The architecture can be seen in Figure 2.2 and is described in much more detail in the original paper [19]. In the following we will highlight the relevant details about the architecture in order to understand our research.

The input is mapped to an embedding, which is learned through a default PyTorch embedding layer. Subsequently, the embedding vector is binarised using the same arctangent surrogate function as of the Spikingjelly LIF neuron to enable backpropagation. The number of time steps used in the network is the length of the input sequence, which is at most the context length long. This form of encoding is not biologically inspired.

Through the linear attention mechanism, SpikeGPT can be formulated as transformer during training and RNN during inference. Instead of a traditional attention block followed by a feed forward network for each layer, an attention free Spiking RWKV (SRWKV) module is used followed by a Spiking Receptance Feed-Forward Network (SRFFN). In both modules, LIF neurons from spikingjelly with $\alpha = 2$ are used as activation functions at the end of each SRFFN or SRWKV module. Some experiments in the original paper are run by swapping LIF neurons with $ReLU$ squared functions. Each SRFFN module consists of three dense layers with a feed forward dimension

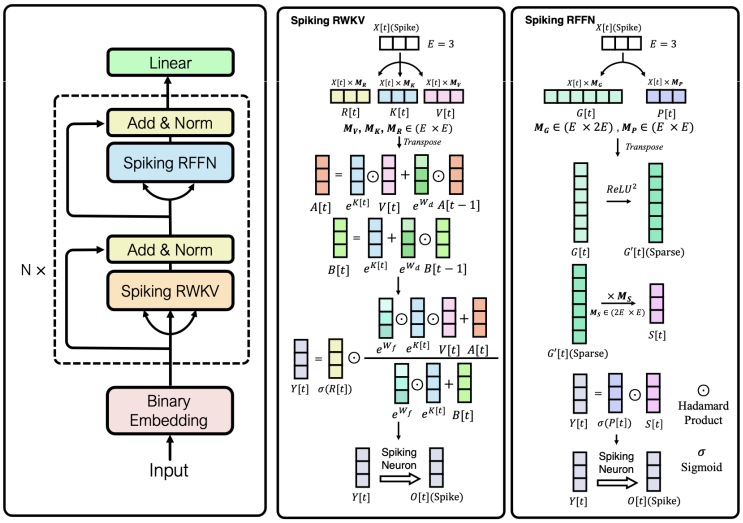

Figure 2.2: Architecture of SpikeGPT. The picture is adopted from Zhu *et al.* [19] with permission from the authors.

of 4 times the embedding size. The standard SpikeGPT model can be run in the default RWKV mode or the RWKV-ffnPre mode. In the RWKV-ffnPre mode, the attention block of the first layer consists of an RWKV channel mix instead of time mix. The dropout is applied at the end of each SpikeGPT block. There is no embedding dropout.

In the following we will explain the advantages of the SpikeGPT architecture, which make it perfectly suitable for answering our research question. The transformer-like formulation of SpikeGPT allows for fast and parralelizable training, which is not possible with RNNs. Additionally, the formulation as recurrence during inference allows for linear complexitiy during inference with a theoretically infinite context length, in contrast to the quadratic complexity of transformers during inference, limiting them to short context lengths. The linear complexity during inferences, makes SpikeGPT also a promising candidate for longer sequences. Coming with these advantages, we want to compare SpikeGPT to state-space models (SSMs) [48] (an easy to understand explanation of SSMs can be found at Grootendorst [49]) which are in the same class of architectures and have recently shown great success in de novo design [36]. SSMs are also made for sequence and signal processing, offering the same advantages through parallelizable training by modulating state spaces as convolution during training and modulating state spaces as recurrence during inference. Due to the binary input and LIF neurons used in SpikeGPT, we expect to have matching or slightly worse performance than [36], while significantly reducing the computational complexity and energy consumption.

2.8 Intel Loihi 2

The Intel Loihi 2 processor is the second generation of the Intel Loihi [50], which is a neuromorphic processor specifically made for modelling biologically plausible neurons used for example in Spiking Neural Networks in silicon. Instead of separating storage and computational unit, neuromorphic hardware works similar to the brain by doing in memory computing. Only through this specific hardware architecture, it is possible to realize the latency and energy efficiency increases of spiking neural networks. The Intel Loihi 2 can be accessed over the intel neuromorphic research cloud (INCR) which is only open to members.

Chapter 3

Methods

3.1 Virtual Screening

3.1.1 Dataset

From the MoleculeACE benchmark [51], we select the dataset of molecules, of which the bioactivity is indicated towards the dopamine D3 receptor(D3R) (ChEMBL ID CHEMBL234). The D3R is often a target for medication that aims at treating substance abuse [52]. This dataset contains 3657 molecules from CHEMBL v29 [53] represented as canonical SMILES strings and is already divided into a training set of size 2924 and test set of size 733. In order to train our networks using early stopping on the validation loss, we split up the training set in a training and validation set with a stratified 80-20 split. The authors of MoleculeACE performed standard pre-processing methods of the SMILES strings, to which we did not make any additions or modifications. For each SMILES string, the bioactivity towards D3R is given using the inhibition constant K_i. K_i represents the concentration of the molecules (in nanomoles [nM]) required to occupy 50% of the D3R. A lower K_i value is more desired, as it indicates a stronger binding affinity towards the D3R.

In Van Tilborg *et al.* [51], regression was performed on predicting the logarithmic K_i values, but we decide to transform the problem into a binary classification task. The reason is the usage of the lava-dl library, which was too complex for us to use for regression due to many issues in the library itself and lack of documentation or support. To convert define a binary classification task, we divide the molecules into active and inactive molecules towards the D3R, based on their K_i values (see 3.1). A threshold of 100 is commonly used in virtual screening, where most data sets contain more

inactive than active molecules towards the target. Hence, every molecule with a K_i value below the threshold of 100, becomes an active molecule and every value equal and above an inactive molecule towards that target. This creates 1885 active and 1039 inactive molecules in the initial training set and 490 active and 243 inactive molecules in the test set.

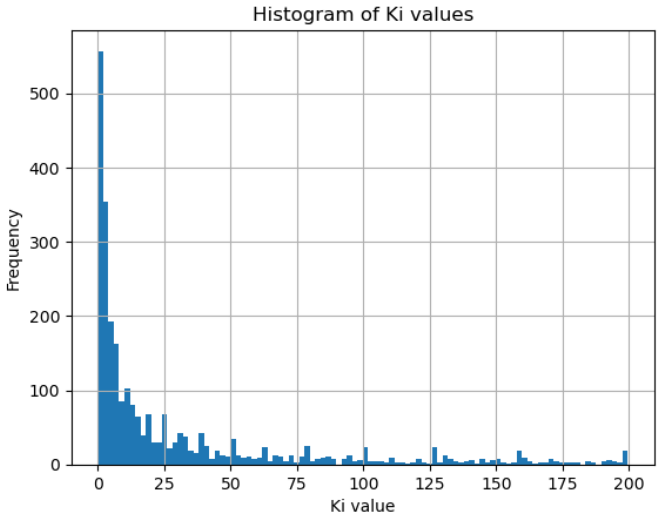

Figure 3.1: Distribution of K_i values for K_i values smaller than 200. Larger K_i values are omitted for readability, as there is no significant increase in occurrence of larger K_i values.

The dataset has already been used for extensive benchmarking for bioactivity prediction using a wide variety of conventional model architectures. A small dataset size is convenient for experimenting with a lot of hyperparameters. It is important to mention that the dataset is made to benchmark the potential of ANNs for handling activity cliffs, which gives our SNN a very hard time. Activity cliffs are molecules with similar structures that are active against the same target, but with great difference in potency (dose of molecule required to produce a pharmacological effect the desired intensity).

3.1.2 Molecule Encoding

In order to use the SMILES strings of our dataset in SNNs, we convert them into a binary representation by first transforming them into binary fingerprints and then duplicating the fingerprint along an additional temporal dimension to create a rate encoding, as done in Nascimben and Rimondini [54]. In our experiments we use Morgan fingerprints with a size of 1024 and radius of two, as well as MACCS keys with a length of 167. As there is no theoretical reasoning to pick a certain number of time steps and the number

of time steps used in Nascimben and Rimondini [54] is not reported, the temporal dimension is subject to hyperparameter search.

Molecular Fingerprints show one of the best performances among molecular representations used in virtual screening [55], also for the dataset we use [51]. In comparison to SMILES strings, they allow us to work with much smaller models and datasets, as they directly encode chemical information. They can be rapidly computed [56] and are therefore already used for screening libraries with billions of molecules [10, 57], making them a suitable fit for exploring more of the chemical universe. Furthermore, we can directly feed fingerprints to SNNs without conversion, as has been shown in a successful application of SNNs for toxitivity prediction [54].

We choose MACCs and Morgan fingerprints because due to their size difference, we can discuss potential trade-offs between performance and memory footprint.

Due to the advantages of SNNs in dealing with sparse data, we looked at different Morgan fingerprint sizes and varying radius for our dataset, to determine the best fitting one for our problem (see table 3.1). Sparsity is computed by taking the average number of 0s per sample and then expressing it as ration of the fingerprint size. A radius higher than 3 is not really used in literature, as it creates too large substructures. We can see that trading a bit of sparsity, we can use half the fingerprint size. Increasing the radius, does not have a significant impact on the number of collisions. That is why we decide on Morgan fingerprints with size 1024 a radius 2, which is also very commonly used in virtual screening.

Table 3.1: Number of collisions and sparsity of MACCS keys and Morgan fingerprints with different sizes and radii.

Fingerprint	Size	Radius	Collision	Sparsity
MACCS keys	167		717	0.676
Morgan fingerprint	1024	2	22	0.949
Morgan fingerprint	1024	3	9	0.929
Morgan fingerprint	2048	2	22	0.974
Morgan fingerprint	2048	3	9	0.964

3.1.3 Spiking Neural Network Architecture

For predicting whether the molecules bind with the dopamineD3 receptor(D3R) we build our own spiking neural network architecture in lava-dl.

The architecture consists of three current based leaky integrate and fire (CUBA) `Dense` blocks from the lava-dl library. Each neuron has a number of parameters that are tuned during hyperparameter search. The threshold determines the required synaptic integration potential for a neuron to fire. The current and voltage decay indicate the synaptic integration potential and membrane potential decay rate respectively. We can also set a time constant (referred to as `tau grad` in lava-dl) for the spike function derivative that determines how quickly the neurons' membrane potential changes in response to synaptic input. The spike function gradient can also be scaled (referred to as `scale grad` in lava-dl) to control the gradient flow across layers. The scaling should be increased if there is a vanishing gradient and decreased if there is an exploding gradient. Furthermore, we can decide whether we want to enable learnable voltage and current decays instead of constant ones (using `require grad`). Last but not least, we can set a flag to enable weight normalisation and to introduce axonal delay. This means, introducing a delay between the spike generation and activation of the synapse. In our human brain, this delay is always present due to the limitations of physics. The network features three CUBA blocks to have a small architecture for our prototype. To prevent overfitting, we use a dropout between the first and second layer and second and third layer. The dropout for SNN works similar to `torch.nn.Dropout`. However, dropout over time dimension is preserved, i.e. if a neuron is dropped, it remains dropped for the entire time duration. The remaining neuron parameters are determined through a hyperparameter search. The number of input neurons is determined by the size of the fingerprint used and the number of output neurons is two, corresponding to the two classes we want to predict. A prediction is made from the two output neurons, by choosing the class of the neuron with the highest firing rate for a given input.

During training, we use the `slayer.loss.SpikeRate` loss function, which determines the loss $\mathcal{L}$ by computing the mean over the mean square error for each neurons' target spike rate (TR) and actual spike rate (SR) (see 3.1). In the case of our binary classification task (with two classes C) we have the following example: Lets say we have a true rate of 0.8 and false rate of 0.03. Then for the output neuron that corresponds to determine how certain the model is that a molecule binds to the target, should have a firing rate of 0.8 if it believes that the new sample binds to the target. The same for the other class.

$$\mathcal{L} = \frac{1}{C}\sum_{j=1}^{C}(SR_j - TR_j)^2 \tag{3.1}$$

We directly train our SNN using SLAYER 2.0 and either the Adam or SGD

optimizer.

We build our spiking neural networks in lava-dl, as it is the only software development framework that is well-developed for directly running SNNs on neuromorphic hardware without converting them to another library [42]. This is furthermore supported by Intels engagement in the Loihi project, through its recent release of the Intel Hala Point Neuromorphic System [58], which is the largest neuromorphic processing unit to date.

The SNN architecture is influenced by various factors. Simple dense layer ANN networks gave the best results in the MoleculeACE benchmark and dense SNNs gave comparable to state-of-the-art results in Nascimben and Rimondini [54]. Furthermore, due to lava-dl we have a limited choice of which SNN architecture to choose, without extending the library. We also tried to evaluate LSTM like SNNs, but failed implementing them due to bug # 285 in the lava-dl library. We use LIF neurons, as they are the fastest neurons and most widely used [17].

In lava-dl there are different options available for training our SNN. We directly train the SNN using `lava.lib.dl.slayer`. Alternatively, we could also select the bootstrap algorithm, which we verified to be faster. However, due to the theoretical performance limitation of doing ANN to SNN conversion, we decide against this algorithm. We first want to see whether we can outperform the ANN.

3.1.4 Artificial Neural Network Architecture

The Artificial Neural Network designed to be comparable to our SNN consists of multi-layer-perceptron with three dense layers, intertwined by two ReLU activation functions. Between the first and second and second and third layer, a dropout is used that is set during hyperparameter search. The network is trained using backpropagation, with a cross entropy loss function. The choice of the optimizer is suspect to hyperparamter search.

3.1.5 Hyperparameter Tuning

For each experiment we use Optuna [59] with a sampler using the TPE (Tree-structured Parzen Estimator) algorithm to minimize the validation loss by optimizing the hyperparameters for 100 trials on one NVIDIA RTX-2080 Ti. For reproducability we use a manual seed when initialising each model. Only 100 trials are performed due to the low impact of hyperparamters in SNNs reported in Nascimben and Rimondini [54] and the low number of hyperparameters for our ANN. To investigate the impact of the number

of time steps, we conduct a separate hyperparameter search for each SNN and fingerprint using time steps of 2,4,8,16 and 32, as well as a separate experiment where we include the number of time steps between 1 and 25 as hyperparameter in optuna. During each trial, we use 5-fold cross validation with our train-validation split, but without oversampling as in Nascimben and Rimondini [54], due to having less class imbalance. Each fold uses the same hyperparameters and is trained for 100 epochs with a batch size of 64, before reporting the mean validation loss across all folds to Optuna. In each epoch we use early stoping criterion on the validation loss with a tolerance of 10^{-5}. The search space is defined in Table 3.2.

Table 3.2: Hyperparameter space used for each ANN and SNN experiment. The meaning of each neuron parameter is explained in Section 3.1.3.

Hyper-parameter	SNN	ANN
Number of hidden neurons	$512 + 128 \cdot n$, $n \in [0, 12]$	$512 + 128 \cdot n$, $n \in [0, 12]$
Learning Rate	$(10^{-5}, 0.5)$	$(10^{-5}, 0.5)$
Optimizer	Adam, SGD	Adam, SGD
Dropout	$0.0, 0.05, 0.1$	$0.0, 0.05, 0.1$
Threshold	$(0.0, 2.0)$	-
Current decay	$(0.0, 1.0)$	-
Voltage decay	$(0.6, 1.0)$	-
Tau grad	$(0.0, 1.0)$	-
Scale grad	$(0.5, 4)$	-
Require grad	True,False	-
Weight normalisation	True, False	-
True Rate	$(0.8, 1.0)$	-
False Rate	$(0.0, 0.2)$	-
Delay	True, False	-

For the number of hidden neurons, batch size and the learning rate, we oriented ourselves at the search spaced used in Nascimben and Rimondini [54]. The optimizer candidates are selected based on commonly used choices in deep learning for our task. For the neuron parameters, we make the search space as large as our computational budget allows due to missing heuristics on what values to choose and orient ourselves at values used for SNNs implemented in lava-dl for other tasks.

3.1.6 Testing

For each experiment, we benchmark the model after hyperparameter search with the following procedure. Using the best hyperparameters found, we re-train the ANN or SNN on the entire training set for the number of epochs the best model was trained on during hyperparameter search. Next, we measure the balanced accuracy, recall, specificity, ROC curve and AUC score of the model on the test set. The balanced accuracy is computed to take into account class invariance and is defined as

$$\text{Balanced Accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2}$$

Confidence values used for ROC curves and AUC scores are computed by applying the softmax function on the raw output of the network (also called logits).

3.1.7 Training Efficiency

We measure the time and energy required for training the best performing ANN and SNN. For each representation we train the ANN and SNN on the entire train set using the same training code as during testing. We measure for each training run the time it takes in seconds and the amount of energy consumed by one NVIDIA RTX-2080-Ti in joules after 100 epochs. Each training run is repeated 10 times and the average time and energy consumption is computed. The time and GPU energy consumption of each training run is measured using Zeuss [60], a library specifically made for measuring energy consumption of deep learning.

3.1.8 Inference Efficiency

For our ANNs and SNNs we use the following procedure to measure the energy per inference in mJ and the number of predictions per second. For the first 10 SMILES strings in our dataset, we perform 6250 predictions for each smile using a batch size of 1. First we used 625 predictions, similar to reference code for pilotnet, but due to loihi warnings that there are not enough measuring points (due to our higher speed than the example notebook), we increased number of steps to 100,000. Due to the practical limitations of Loihi 2 we can only choose SNN models with time steps of powers of 2. We measure the total amount of power consumed by the loihi processor, without subtracting the static power it consumes without computations. Execution time and power consumption is measured using intels `loihi2_profiler` class, which is recommended by intel for measurements. For the ANN we use Zeuss to measure power and latency on a NVIDIA RTX-2080-Ti.

3.2 De novo design with SpikeGPT

In order to investigate the potential of SpikeGPT for de novo design, we start with the most fundamental task in de novo design. Teaching the model the 'chemical language' and generating unique molecules that are not part of the training set. To benchmark SpikeGPT against the state-of-the-art, we choose our methodology in a way to make a direct comparison to the S4, GPT and LSTM model of the work presented in Özçelik *et al.* [36]. This means, we use the same dataset, model size, hyperparameter search strategy and evaluation metrics as in Özçelik *et al.* [36].

3.2.1 Dataset

Our training and validations set consists of 1,900,000 and 100,000 SMILES strings from the ChEMBL database. The SMILES strings are sanitized, canonicalized, label encoded and then padded to a length of 100. This is done by appending an end and start token before and after the SMILE and adding padding tokens until a length of 100. Furthermore, the SMILES strings are fed in random order to our model.

3.2.2 Architecture

During training, we use the default SpikeGPT code from Zhu *et al.* [19] integrate it into the code base from Özçelik *et al.* [36] to use SpikeGPT for de novo design. We use SpikeGPT in the default RWKV mode instead of the RWKV-ffnPre mode, as the RWKV-ffnPre mode has been shown to have almost no change in behaviour in the original SpikeGPT paper. Each model is initialised with the same manual seed for reproducibility and unless specifically mentioned, we use a context length of 100 to match the default maximum SMILE length we use.

Due to the non-biological inspired binary encoding used in the SpikeGPT paper, we looked at limited alternative methods used in the field of SNNs and came to the conclusion that we do not change the default encoding. One research [61] created a rate encoding from word embeddings obtained through Word2Vec [62]. For each value of the embedding, a poisson spike train with an arbitrarily chosen time dimension is created, with a firing rate corresponding to the value. Motivated by the results of SpikeGPT for language modelling and the large model and data size in de novo design, we decide to avoid an additional temporal dimension. Another paper by researchers at TU Graz and Intel [63] used a threshold crossing encoding method, which is described in detail in their paper. This encoding however, is

directly coupled to the number of neurons used and the network architecture, which would require us to change the SpikeGPT architecture. Due to time limitations, we were not able to verify whether it would be possible to use this encoding.

Given an input sequence of n SMILES tokens $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, our model predicts the next token s_{n+1} of the sequence.

3.2.3 Training Loop

In our training loop the last token is split off from each sample to generate an input and label token the model has to predict. The loss function is computed by taking the cross-entropy between prediction and label. By default, an Adam optimizer is used to train the model with the same epsilon of $4e-9$ and beta coefficients $(0.9, 0.9)$ as used in Zhu *et al.* [19]. We disabled learning rate decay due to our much smaller dataset size, as well as for comparability with the S4 paper and to keep things simple for a first attempt.

3.2.4 SMILES Generation

When generating SMILES strings, we start from scratch (de novo), by giving the model a beginning token and letting it predict the next token. The next token is determined via temperature sampling with a temperature T, from the probability distribution of each token. Then, feeding the entire sequence into the model again and letting it predict the next token. This procedure is repeated until we generated 100 tokens. As we are not using our model as chatbot, we changed the RWKV_RNN model to support batch generation of molecules.

In order to benchmark how well SpikeGPT learns the SMILES syntax, we use a set of metrics, which are also used in the S4 paper. Validity is computed by the number (and frequency) of SMILES strings corresponding to chemically valid molecules, following the judgement of the `Chem.MolFromSmiles` function in rdkit. Novelty measures the number (and frequency) of unique and valid designs that are not included in the training set. Uniqueness quantifies the number (and frequency) of structurally unique and valid molecules among all designs. We could have used more metrics, such as from the Moses benchmark [64], but these metrics have their own issues, letting us believe that validity, uniqueness and novelty is enough to assess the ability of SpikeGPT to learn the chemical language.

3.2.5 Hyperparameter Optimization

Our hyperparameter search space is largely inspired by the works of Özçelik *et al.* [36]. For the model size, their paper uses model sizes around 850k to 6.6M learnable parameters, while we use models with 1.7M to 6.8M learnable parameters, for comparability. Furthermore, we try the PyTorch default ϵ value of 10^{-8} and the SpikeGPT default of 410^{-9}. Even though it is not explicitly mentioned in neither the SpikeGPT nor RWKV paper, we assume that a smaller epsilon is choosen to have a larger variance in the learning rates when using adaptive learning rates. As we use a constant learning rate, we evaluate both values to be on the save side. Following the strategy of Özçelik *et al.* [36], we perform a random search over our hyperparameter space (see Table 3.3). Training is terminated early based on no decrease in the validation loss for 5 epochs with a tolerance of 1×10^{-5}. We do not perform cross validation due to computational complexity of our models. For each model, we generate 10240 SMILES strings de novo with a batch size of 1024 and T=1. Afterwards we calculate validity, uniqueness and novelty. Using Raytune, the hyperparameter search was conducted on six NVIDIA RTX-3090 with 24 GB of VRAM each in parallel. Each trial used 2 CPUs and 6 GB of RAM. In total we ran 200 different trials with training times ranging from 1h to 32h per trial.

Table 3.3: Hyperparameter search space used to optimize the SpikeGPT model for de novo design.

Hyper-parameter	Search Space
# SpikeGPT Blocks	$2, 3, 4, 5, 6, 7, 8$
Embedding size	$32, 64, 128, 256$
Dropout rate	$0.0, 0.1, 0.2$
Batch size	$512, 1024$
Learning rate	$(10^{-6}, 0.1)$
eps	10^{-8}, 4×10^{-9}

In the following we will describe how we will determine our best SpikeGPT model, that we compare against S4, LSTM and GPT. We take the 10 runs of the hyperparameter search with the highest novelty. Then we sample for each model $10,240$ SMILES strings for different sampling temperature (T) values from $T = 1$ until $T = 2$, with a step of 0.25. For each generation we benchmark validity, uniqueness and novelty. Afterwards, we take the model with the highest novelty.

3.2.6 Efficiency Evaluation

As the implementation of SpikeGPT on the Intel Loihi 2 by a research group at the University of California, Santa Cruz is not completed by the end of this research, we could not experimentally benchmark the efficiency of SpikeGPT during inference. Therefore, we can only refer to the theoretical analysis performed in Zhu *et al.* [19]. However, for training we measure the time and energy consumed by SpikeGPT using Zeuss compare it to the S4 model. Each model is trained with eight S4 or SpikeGPT blocks, a learning rate of 0.01, a dropout rate of 0.1 and batch size of 1024. Then, we train each model for 100 epochs on the entire train set. All experiments are performed on an NVIDIA A100 80GB GPU using the standard torch optimizer (without deepspeed for SpikeGPT).

Chapter 4

Results

Trying to close the outlined research gap of how to explore more of the chemical universe when using machine learning to identify hit candidates, we present our results of using SNNs for virtual screening (Section 4.1) and de novo design (Section 4.2) and compare them to traditional deep learning methods.

4.1 Virtual Screening

First, we use a SNN for screening a database of molecules by predicting the bioactivity of molecules towards the dopamine D3 receptor. This task is a binary classification task, where the Spiking or Artificial Neural Network needs to decide whether a molecule will be bioactive towards the dopamine D3 receptor. The best classifier of each architecture using Morgan fingerprints or MACCs keys is evaluated for its balanced accuracy, recall, specificity and AUC score in Table 4.1. In general, the SNN is able to deliver comparable or better results than the ANN across all measured metrics. Even though the ANN has a higher balanced accuracy (except for MACCS keys), recall and AUC score than the SNN, the SNN is more accurate at predicting inactive molecules towards the dopamine D3 receptor (measured by the specificity). Looking at the different fingerprints, Morgan fingerprints give a higher balanced accuracy for both models, but SNNs are better at predicting bioactive molecules correctly when using MACCS keys instead of Morgan fingerprints (measured by the recall). Furthermore, SNNs have a higher balanced accuracy than ANNs when trained on MACCS keys, but lower AUC score.

Table 4.1: Balanced accuracy, recall, specificity and AUC score of the best ANN and SNN for predicting bioactivity of molecules towards the dopamine D3 receptor after being trained on MACCS keys or Morgan fingerprints.

Fingerprint	Size	Model	Balanced accuracy	Recall	Specificity	AUC
MACCS keys	167	SNN	78.6 %	0.849	0.724	0.83
MACCS keys	167	ANN	78.4 %	0.873	0.695	0.87
Morgan fingerprint	1024	SNN	81.2 %	0.822	0.802	0.87
Morgan fingerprint	**1024**	**ANN**	**81.7%**	**0.897**	**0.736**	**0.91**

As the number of time steps used in the SNN is subject to hyperparameter tuning, we investigate the balanced accuracy and AUC score of different number of time steps (see Figure 4.1). This allows us to reason about trade-offs between accurately predicting bioactivity and reducing the number of time steps to improve memory consumption and processing time, to subsequently explore more of the chemical universe. For each separate experiment with a fixed time dimension between 2 and 32, which is a power of 2 to be compatible with the Intel Loihi 2, we report the best model. Additionally, we include for each representation the best model where the time dimension was open to hyperparameter search. All SNNs perform better than random guessing when using more than 8 time steps and only fluctuate slightly for larger temporal dimensions. More detailed results on the impact of the temporal dimension on recall and specificity can be found in Chapter A.

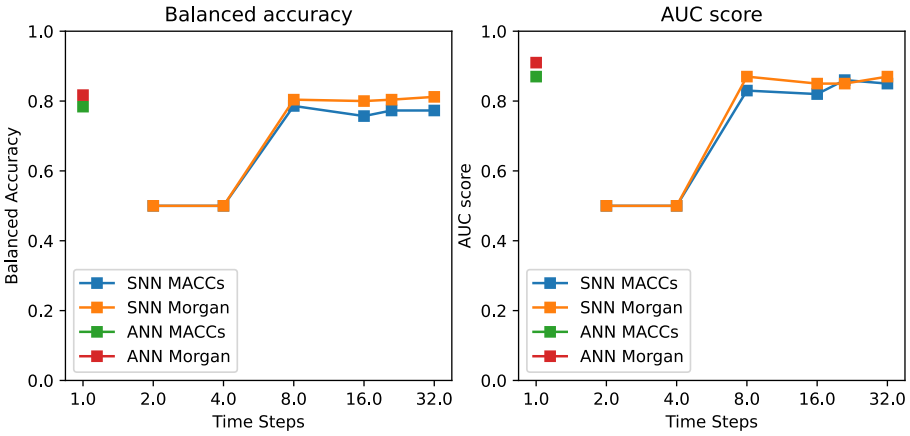

Figure 4.1: Balanced accuracy and AUC score of the SNN using MACCs or Morgan fingerprints for time steps between 2 and 32 in comparison to the ANN.

After demonstrating that SNNs can match the performance of ANNs for bioactivity prediction, we evaluate their ability to efficiently explore the chemical universe in comparison to ANNs. Hence, the time and energy required during training is analyzed in Table 4.2, as well as the number of predictions per second and the amount of energy per prediction in Table 4.3. Trading off a slight amount of balanced accuracy for the SNN using Morgan fingerprints, we benchmark the best SNN with 8 time steps for each molecular representation. In general, Morgan fingerprints allow for faster computation times and lower energy consumption. During training, the ANN is clearly ahead of the SNN by being on average 2.9-4.5 times faster, while consuming 4.3-5.6 times less energy. However, the tables turn when evaluating the classifier for predicting bioactivity of unknown molecules. The SNN manages to perform on average 2.7-4.1 times more predictions, while using 100 times less energy for MACCS keys and 200 times less energy for Morgan fingerprints. Combining both stages, we can say that the SNN starts saving energy over the ANN after 951,576 predictions (taking 42.8s) when trained on MACCS keys and 603,578 predictions (taking 15.8s) when trained on Morgan fingerprints. Additionally, the SNN starts making more predictions than the ANN after around 58.9s for MACCS keys and 17.3s for Morgan fingerprints.

Table 4.2: Average time and energy consumed when training our ANN or SNN on different fingerprints.

Fingerprint	Model	Avg time [s]	Avg energy [J]
MACCS keys	ANN	28.83	1781.00
MACCS keys	SNN	130.98	10039.73
Morgan fingerprint	**ANN**	**27.8**	**1780.19**
Morgan fingerprint	SNN	82.2	7624.64

Table 4.3: Average number of predictions per second (PPS) and energy consumption per inference (EPI) when predicting bioactivity using our ANN and SNN.

Fingerprint	Model	Hardware	PPS	EPI [mJ]
MACCS keys	SNN	Loihi 2	22,187	0.081
MACCS keys	ANN	GPU	8121	8.76
Morgan fingerprint	**SNN**	**Loihi 2**	**38,009**	**0.047**
Morgan fingerprint	ANN	GPU	9210	9.73

4.2 De novo design with SpikeGPT

To explore Spiking Neural Networks for de novo designs, we compare SpikeGPT against the measurements of the S4, GPT and LSTM model in Özçelik *et al.* [36] for its ability to learn the chemical language. Hence, we let SpikeGPT generate 102400 molecules de novo and measure the amount of valid, unique and novel SMILES strings in comparison to the other models (see Table 4.4). With a validity and uniqueness of 93 %, SpikeGPT performs better than a normal GPT model, but falls short of the validity on an LSTM or S4. Nevertheless, SpikeGPT ranks second place together with GPT, when it comes to novelty.

Table 4.4: Comparison of validity, uniqueness and novelty between SpikeGPT, S4, LSTM and GPT. The measurements of the S4, LSTM and GPT model are adopted from Özçelik *et al.* [36].

Model	Valid	Unique	Novel
SpikeGPT	95,760 (93%)	95,720 (93%)	91,390 (89%)
S4	**99,268 (97%)**	**98,712 (96%)**	**95,552 (93%)**
LSTM	97,151 (95%)	96,618 (94%)	82,988 (81%)
GPT	93,580 (91%)	93,263 (91%)	91,590 (89%)

For both models we analyze the invalid molecules that it generates and group them based on different error categories in figure 4.2. We distinguish between errors made due to wrong ways of molecules branching out into other molecules, the way rings are represented, chemical bonds are assigned and other (miscellaneous) syntax errors. Even though SpikeGPT makes the third-largest amount of syntax errors and largest amount of branching errors, it is much better than the LSTM and GPT model at avoiding ring errors.

Due to time constraints, we only benchmark SpikeGPT more extensively against the original S4 model that we obtained from the authors of Özçelik *et al.* [36]. To test the robustness of the model, we sample 10240 molecules with a batch size of 1024 by our best SpikeGPT model for sampling temperature (T) values between 1 and 2 with a step of 0.25 and measure validity, uniqueness and novelty (see Figure 4.3). We can see that for both models, all metrics decrease with increasing T values. Nevertheless, S4 remains much more robust for larger T values.

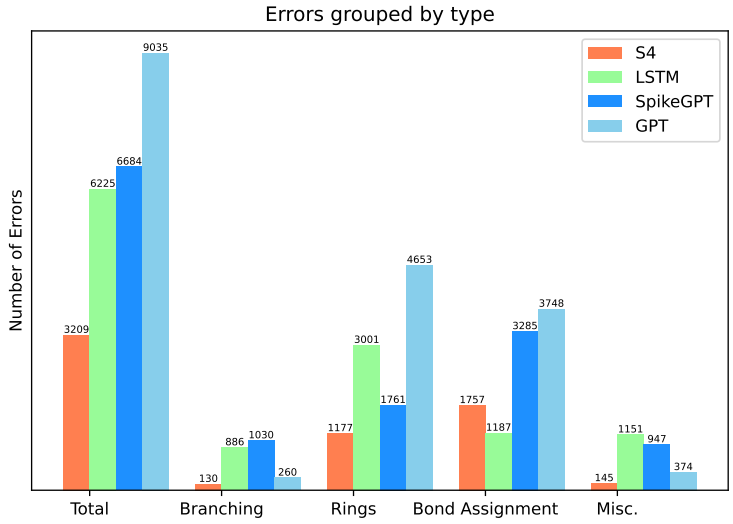

Figure 4.2: Syntax errors grouped by type after generating 102,400 molecules with SpikeGPT,S4, LSTM and GPT. The measurements of the S4, LSTM and GPT model are adopted from Özçelik *et al.* [36].

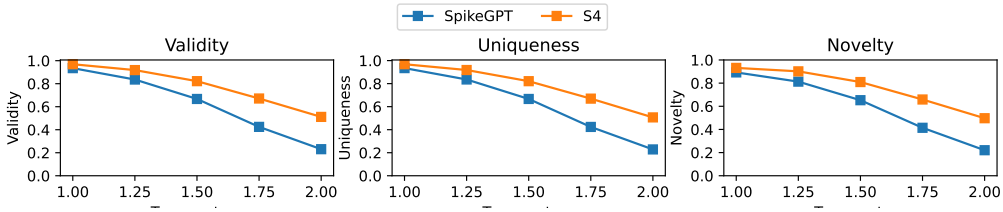

Figure 4.3: Validity, uniqueness and novelty when generating 10,240 molecules for T values between 1 and 2.

As described earlier, we did not manage to run SpikeGPT on neuromorphic hardware. Therefore, we only provide results for the time and energy consumed per epoch by SpikeGPT in comparison to S4. SpikeGPT takes less than 8 min per epoch, consuming 0.13 MJ, while S4 takes more than 17 min per epoch, consuming 30.52 MJ. This is a time difference with a factor of around 2 and an energy difference with a factor of around 234.

Chapter 5

Discussion

Our initial hypothesis was that SNNs match the performance of ANNs in drug discovery or perform slightly worse, but instead offer significant improvements in latency and energy consumption during inference to explore more of the chemical universe. Through our experiments (Chapter 4), we can validate this hypothesis for virtual screening but are unable to for de novo design due to missing experiments of SpikeGPT on neuromorphic hardware.

5.1 Virtual Screening

One important point of discussion for our virtual screening experiments is the introduction of an additional temporal dimension because of the rate encoding that SNNs use. Even though our SNNs is significantly faster and consumes less energy than the one dimensional fingerprints used in ANNs, we have not measured and compared the concrete memory overhead of SNNs in bytes. Instead, this research gives energy and latency measurements on the impact of varying number of time steps. For both representations, we see that the SNN performs random guessing until the number of time steps exceeds 8 time steps. Larger temporal dimension do not result in drastic improvements in overall balanced accuracy or AUC score. In the case of MACCS keys, shorter time steps seem to give the best results. This lets us come to the conclusion that depending on the importance of prediction accuracy, one could for example trade-off 4 times less memory consumption of each sample for 0.8% less balanced accuracy in the case of Morgan fingerprints. Using Morgan fingerprints for our SNNs results in about 3 % higher balanced accuracy as well as AUC score, while also being faster and less energy demanding during training and inference. We explain the difference in performance through the larger fingerprint size, which results in lower numbers of collisions of

fingerprints in comparison to MACCS keys (see Figure 3.1). Intuitively, the smaller MACCs keys should be faster and less energy demanding during their smaller size, however the works of Szafarczyk *et al.* [65] show that MACCS keys have longer computation times than Morgan fingerprints. Following the methodology used in lava-dl tutorials to measure inference latency and energy, we measured a significant improvements of SNNs over ANNs. However, we want to give caution for drawing too drastic conclusions from this result as we only use a batch size of 1, which is never used in practice. Considering all the pros and cons of our SNN for bioactivity prediction, we believe that SNNs will be most useful for screening ultra large libraries (such as ZINC [9]) with billions of compounds after training them once on a large bioactivity datasets such as the LIT-PCBA dataset [66] with at most around 350,000 molecules for a single molecular target. Only then, the speed and energy gains during inference are worth the small loss in performance. Due to the difficulties of theoretically evaluating bioactivities of molecules accurately and high lab experiment costs, bioactivity datasets are traditionally tiny (less than 500k molecules) in comparison to the chemical universe (up to 10^{60} molecules). This could be an issue for the generalisation performance of the model when screening billions of molecules due to relatively small training set. To our knowledge there is no existing research exploring this relationship. One way to avoid this potential issue is by integrating our model with automatic synthesis labs that do not require human labour, such as the A-lab [67]. After the model has determined a set of hit candidates, a small amount could be synthesised automatically by the lab to evaluate their bioactivity towards the target experimentally and afterwards increase the size of the training set. Then the whole process can be repeated as many times as necessary.

5.2 De novo design with SpikeGPT

SpikeGPT shows great promise for matching the state of the art in chemical language modelling, when it comes to learning the SMILES syntax. Even though it does not surpass the best model (S4) in any metrics, the results are close enough for us to conclude that these trade-offs in performance might be worth it for potential improvements in efficiency. A limitation that our research can not address is that despite the potential to generate much more of the chemical universe, we can not ensure chemical synthesizibility of the generated molecules, which is also an issue for most other models [68]. One interesting finding for us is that SpikeGPT delivers comparable results to other architectures despite a small context length of 100 in comparison to the context lengths between 1024 and 3072 used in Zhu *et al.* [19]. As the original paper of SpikeGPT reports significant reductions in performance

for context lengths of 1024 and the sequence length represents the temporal dimension of the SNN, which we have seen in Chapter 3.1 to be highly influential to the performance of the classifier. Our speculation is that we do not observe the same effect due to starting generation from scratch each time. In the beginning we were worried about the small prompt size as the RWKV model that SpikeGPT is based on reports a high sensitivity of RWKV to prompts [47]. However, we decided to not further investigate this issue as Zhu *et al.* [19] does not report such sensitivity. When looking at efficiency, we unfortunately did not manage to run SpikeGPT on neuromorphic hardware. Therefore, we can only speculate that SpikeGPT will take a lot less time and energy during generation on neuromorphic hardware. This is based on the theoretical estimation for large language modelling in Zhu *et al.* [19] and our experiments for virtual screening. Furthermore, we need to highlight the unfair comparison between S4 and SpikeGPT during training. SpikeGPT uses a custom CUDA kernel and deepspeed [69] for faster training. Nevertheless, we include this experiment to give the reader an approximate estimation on the computational complexity of SpikeGPT during training. Last but not least, we want to discuss a potential use case of SpikeGPT in drug discovery. Similar to SNNs in virtual screening, we believe that SpikeGPT will be most useful for foundational models that make more than one million predictions.

Chapter 6

Related Work

Using manual literature search and Litmaps, we are only aware of the works of Nascimben and Rimondini [54] for using SNNs for drug discovery. In comparison to Nascimben and Rimondini [54], which shows comparable results of SNNs and ANNs for toxitivity prediction, we are able to verify the same for bioactivity prediction. In contrast to their work, we run our SNN on neuromorphic hardware, allowing us to give a concrete experimental analysis of the efficiency of SNNs over ANNs for drug discovery. Another novelty is the benchmark of different number of time steps in our SNN, giving more understanding on the relationship between memory consumption of SNNs and performance. As expected from the results reported in literature [17, 54], our SNN is between 2.9 to 4.5 times slower to train than a comparable ANN. Our inference latency and energy measurements also match the results reported in literature for other tasks [17, 18, 70]. Additionally, we want to compare our research against other research for screening as much of the chemical space as possible. One research that we considered to draw inspiration from is the deep docking platform [10]. A very simple dense ANN is trained on a million Morgan fingerprints with radius 2 to predict the docking scores of many molecules. The training set is created by computing the docking score for all one million molecules. These predictions are used together in an algorithm, to filter out large parts of the database that are not worth considering as candidate molecules. Through the combination of deep learning and combinatorics, the platform can process billions of molecules in a very fast time. However, we abandoned this direction, as the docking score is shown to be quite unreliable in comparison to real world bioactivity measurements [71, 72].

Continuing with de novo design, we could not find any existing work on using spiking neural networks for this task. Only experimental results can show whether SpikeGPT is faster than S4 at generating molecules, which is

the fastest model in the works of Özçelik *et al.* [36].

Last but not least, we compare our research against related work in the field of more efficient deep learning methods. One class of solutions are ANN accelerators. An example is Amazons recent unveiling of more energy efficient hardware for large language models. However, these accelerators can not overcome the limitations of the van-Neumann bottleneck, which we avoid on neuromorphic hardware. Furthermore, we want to compare our work against different neuromorphic computing platforms. One example which is still a prototype but relatively convenient to try is the IBM Analog Hardware Acceleration Kit [73]. Analog crossbar arrays are used to do training and inference of ANNs in memory, for lower latency and energy consumption. Another example is digital hardware built by Rain Neuromorphic, where memristor arrays are shown to give up to 1000 times more energy efficiency in training than traditional approaches [74]. Their hardware works on 4-bit level, which is not too far away from 1-bit level at which SNNs operate. However, the project is still closed source and not publicly available. There are more similar projects, where in-memory platforms are built to accelerate ANNs. But, we choose against this direction for our research as it is similar in choosing another graphics card for your experiments. An interesting project would be to build specific hardware for drug discovery, which was not possible for me due to my missing engineering background.

Chapter 7

Conclusions

In summary, we have shown that spiking neural networks run on the Intel Loihi 2 allow for exploring the chemical universe much faster and with a lower energy footprint than ANNs using virtual screening. The slight impact of larger temporal dimensions on the accuracy of predicting bioactivity suggests choosing as little time steps as possible, based on the importance of accuracy. We have shown competitive performance of SpikeGPT in comparison to the state of the art for learning the chemical language, but are only able to hypothesis from research on large language modelling, that SpikeGPT will be more efficient for generating molecules when run on neuromorphic hardware. In general, SNNs provide an opportunity to consider more of the chemical universe when finding hit candidates, but come at the cost of slower training times, higher memory footprints when using additional temporal dimensions and slight performance decreases due to the information loss of binary representation. This let us conclude that SNNs are promising for foundational models, that try to explore as much of the chemical universe as possible.

Chapter 8

Future Research

Different SNN architectures could be even more suitable for drug discovery than the ones we used in this research. For example, the architecture of SpikeGPT is much more biologically plausible than traditional ANNs, but to go one step further in becoming more biologically plausible and look into new techniques from our brain to translate to SNNs, we could try the newly published AstroSNN [75] for de novo design. This model implements astrocytes into SNNs, which have a direct impact in the brain on learning and memory. The architecture shows great promise for natural language generation, by having lower latency, higher throughput, and reduced memory usage over SpikeGPT. A reason for these benefits is that it overcomes the longstanding challenge of implementing parallel training processes in traditional neuronal models. We choose the simplest and fasted neuron model (Leaky Integrate and Fire Neurons) due to their fast training times and simplicity. However, future research could look into using more complex neuron models, which have a lot more tunable hyperparameters. One of the biggest limitations for efficiency and performance, are the difficulties of training SNNs. An alternative training algorithm, which is only a proof of concept yet, would be to try the forward propagation training algorithm for SNNs [76], which claims to be able to compute exact gradients instead of surrogate gradients. We consider online learning a promising future direction to look into for de novo design and virtual screening due to the fast online learning capabilities of SNNs [17]. One potential application would be to use foundational SNN models with online learning for identifying hit candidates and integrating them in such a way that they are continuously fine-tuned without re-training by experimental results of automatic labs that synthesize the most promising hit candidates determined by the SNN. Hence, making the best use of the fast online learning and inference capabilities of SNNs. Example online learning techniques are Online Training Through Time [77]

or the online learning algorithms already present for Loihi 2. As the tasks in this research are rather static, we propose to further look into SNNs for molecular dynamics, as the situation is event driven and therefore more naturally suited for SNNs. One example is using neuromorphic hardware for in vitro simulation of alzheimer drugs as part of the Neureka Project [78]. Another direction of future research would be to look into using SNNs for higher dimensional molecular representations. Currently, models are struggling to balance the amount of chemical information encoded, as well as having realistic computational costs [8]. For example, 3D molecular representations are still almost unusable. Graph Spiking Neural Networks [79, 80] could be used to use 2 or 3D molecular information with much more chemical knowledge in a memory and latency friendly way.

When it comes specifically to virtual screening, future research should look into multiple directions to further improve our established methodology. Namely, measuring the impact of different network sizes, the impact of early-stopping, the memory consumption of bytes of SNNs and ANNs when using different numbers of time steps, the robustness of SNNs to more imbalanced datasets and using larger batch sizes for our inference measurements. Another point would be to try out the bootstrap algorithm using ANN to SNN conversion, which has shown significantly faster training times when doing manual tests. With the SLAYER algorithm, we believe that SNNs are only an attractive option for larger foundational models which are only trained once. With the bootstrap algorithm, this could change. Another subject of further research could be the evaluation of the generalisation performance of SNNs with different training sizes, to screen libraries of billions of molecules. Larger bioactivity datasets such as the LIT-PCBA dataset [66] with around 400,000 molecules could be used during this study. Another study for future research would be to incorporate the SNN together with other computational methods into a large scale big data system to process billions of molecules, similar to the Deep Docking platform [10].

Moving to de novo design, there are various experiments we had to leave out due to time and resource limitations. The most significant one being to run SpikeGPT on the Intel Loihi 2, due to the implementation by the University of California, Santa Cruz being still in progress. Equally important is to evaluate SpikeGPT for other de novo design tasks to make conclusions about the promise of SpikeGPT for drug discovery. Examples are generating bioactive molecules, natural product design to generate larger molecules, molecular dynamics simulation and binding free energy calculation. Trying to improve validity, uniqueness and novelty, we could try use pre-trained embeddings from ChemBERTa [81] trained on a much larger SMILES corpus and then applying the same binarization procedure as in the SpikeGPT paper to potentially improve performance. Furthermore, we could try the work

of Wang *et al.* [82] for biologically inspired binary sparse embedding which could make it easier to understand high dimensional chemical representations. Another biologically inspired sparse embedding is Liang *et al.* [83]. When looking at efficiency, we could evaluate sparse embeddings, to investigate the trade-off between efficiency increase and information loss. An important experiment that will show how useful it is to use SpikeGPT for more than a million predictions, will be to test its ability to reconstruct the chemical universe. We envision an experiment similar to [84]. In this paper, a generative model was trained with one million molecules (0.1% of the GDP-13 database) but it was able to reproduce 68.9% of the entire database (containing 975Million molecules) after training, when sampling 2 billion molecules. [84] In the future we also want to more intensively benchmark the impact of the network size, as well as comparing SpikeGPT against more models, such as the vanilla RWKV model. Since we outlined foundational models as the main use case of SpikeGPT in drug discovery, further research should also look into trying out much larger models and dataset sizes. Instead of 8M parameters as in our research, we could go up to 216M parameters as shown in Zhu *et al.* [19].

Data and code availability

All data and code to reproduce and extend the results presented in this thesis can be found at https://github.com/Philipp413/snn-for-drug-discovery. The Intel Loihi 2 is only available to members of the Intel Neuromorphic Research Community.

Acknowledgements

First and foremost, I would like to thank my supervisors, Dr. Twan van Laarhoven (Radboud University, Nijmegen) and Dr. Francesca Grisoni (Eindhoven University of Technology) for their valuable guidance and mentorship throughout the duration of this thesis. I would like to acknowledge Rıza Özçelik (Eindhoven University of Technology) for providing me with his technical expertise and support over the past year. I want to acknowledge the members of the Molecular Machine Learning Team for their constructive feedback and valuable insights, especially during group presentations. My heartfelt thanks to Dr. Jason Eshragian, Rui-Jie Zhu, and the entire UCSC Neuromorphic Computing Group for hosting me for a month and sharing their expertise in Spiking Neural Networks, as well as for their valuable feedback on my project. I am grateful to Dr. Yoeri van de Burgt for his

guidance and knowledge in neuromorphic computing over the past year. For being able to run my experiments on neuromorphic hardware, I want to acknowledge a number of people. I want to thank Dr. Johan Kwisthout for his expertise in neuromorphic computing and allowing me to access the Intel Loihi 2 via the Radboud University in Nijmegen. I am thankful to Intel for providing access to the Loihi 2 as part of the Intel Neuromorphic Research Cloud. Furthermore, I would like to acknowledge Arne Diehl, André Rodrigues Urbano and Dr. Mahyar Shahsavari for their technical assistance with using the lava-dl and the Intel Loihi 2. Last but not least, I express my special thanks to Dr. Juriaan Metz (Radboud University, Nijmegen) and the entire Honors Association of the Faculty of Science at Radboud University for their efforts, guidance, and for making this thesis possible.

Bibliography

[1] R. S. Bohacek, C. McMartin, and W. C. Guida, "The art and practice of structure-based drug design: A molecular modeling perspective," *Medicinal Research Reviews*, vol. 16, no. 1, pp. 3–50, Jan. 1996, ISSN: 0198-6325, 1098-1128. DOI: 10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1098-1128(199601)16:1%3C3::AID-MED1%3E3.0.CO;2-6 (visited on 2024-06-24).

[2] J.-L. Reymond, L. Ruddigkeit, L. Blum, and R. Van Deursen, "The enumeration of chemical space," *WIREs Computational Molecular Science*, vol. 2, no. 5, pp. 717–733, Sep. 2012, ISSN: 1759-0876, 1759-0884. DOI: 10.1002/wcms.1104. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.1104 (visited on 2024-06-24).

[3] *CAS REGISTRY — CAS*, publisher: American Chemical Society. [Online]. Available: https://www.cas.org/cas-data/cas-registry (visited on 2024-06-24).

[4] D. Sun, W. Gao, H. Hu, and S. Zhou, "Why 90% of clinical drug development fails and how to improve it?" *Acta Pharmaceutica Sinica. B*, vol. 12, no. 7, pp. 3049–3062, Jul. 2022, ISSN: 2211-3835. DOI: 10.1016/j.apsb.2022.02.002. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9293739/ (visited on 2023-12-13).

[5] I. Doytchinova, "Drug Design—Past, Present, Future," *Molecules*, vol. 27, no. 5, p. 1496, Feb. 2022, ISSN: 1420-3049. DOI: 10.3390/molecules27051496. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8911833/ (visited on 2023-09-26).

[6] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen, "Innovation in the pharmaceutical industry: New estimates of R&D costs," *Journal of Health Economics*, vol. 47, pp. 20–33, May 2016, ISSN: 1879-1646. DOI: 10.1016/j.jhealeco.2016.01.012.

[7] O. J. Wouters, M. McKee, and J. Luyten, "Estimated Research and Development Investment Needed to Bring a New Medicine to Market, 2009-2018," *JAMA*, vol. 323, no. 9, pp. 844–853, Mar. 2020, ISSN: 1538-3598. DOI: 10.1001/jama.2020.1166.

[8] F. Grisoni, "Chemical language models for de novo drug design: Challenges and opportunities," *Current Opinion in Structural Biology*, vol. 79, p. 102 527, 2023, ISSN: 0959-440X. DOI: https://doi.org/10.1016/j.sbi.2023.102527. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959440X23000015.

[9] B. I. Tingle *et al.*, "ZINC-22—A Free Multi-Billion-Scale Database of Tangible Compounds for Ligand Discovery," *Journal of Chemical Information and Modeling*, vol. 63, no. 4, pp. 1166–1176, Feb. 2023, ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.2c01253. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jcim.2c01253 (visited on 2023-12-02).

[10] F. Gentile *et al.*, "Artificial intelligence–enabled virtual screening of ultra-large chemical libraries with deep docking," *Nature Protocols*, vol. 17, no. 3, pp. 672–697, Mar. 2022, ISSN: 1750-2799. DOI: 10.1038/s41596-021-00659-2. [Online]. Available: https://www.nature.com/articles/s41596-021-00659-2 (visited on 2023-12-02).

[11] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in nlp*, 2019. arXiv: 1906.02243 [cs.CL].

[12] E. Kolbert, "The Obscene Energy Demands of A.I.," *The New Yorker*, Mar. 2024, ISSN: 0028-792X. [Online]. Available: https://www.newyorker.com/news/daily-comment/the-obscene-energy-demands-of-ai (visited on 2024-04-22).

[13] D. Efnusheva, A. Cholakoska, and A. Tentov, "A survey of different approaches for overcoming the processor - memory bottleneck," *International Journal of Information Technology and Computer Science*, vol. 9, p. 151, Apr. 2017. DOI: 10.5121/ijcsit.2017.9214.

[14] I. L. Markov, "Limits on fundamental limits to computation," *Nature*, vol. 512, no. 7513, pp. 147–154, Aug. 2014, ISSN: 1476-4687. DOI: 10.1038/nature13570. [Online]. Available: https://www.nature.com/articles/nature13570 (visited on 2024-06-24).

[15] M. M. Waldrop, "The chips are down for Moore's law," *Nature News*, vol. 530, no. 7589, p. 144, Feb. 2016. DOI: 10.1038/530144a. [Online]. Available: http://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338 (visited on 2024-06-24).

[16] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(97)00011-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117.

[17] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking Neural Networks and Their Applications: A Review," *Brain Sciences*, vol. 12, no. 7, p. 863, Jun. 2022, ISSN: 2076-3425. DOI: 10.3390/brainsci12070863. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9313413/ (visited on 2023-09-26).

[18] Y. Guo, X. Huang, and Z. Ma, *Direct learning-based deep spiking neural networks: A review*, 2023. arXiv: 2305.19725 [cs.CV].

[19] R.-J. Zhu, Q. Zhao, G. Li, and J. K. Eshraghian, *Spikegpt: Generative pre-trained language model with spiking neural networks*, 2023. DOI: https://doi.org/10.48550/arXiv.2302.13939. eprint: 2302.13939.

[20] E. O. Neftci, H. Mostafa, and F. Zenke, *Surrogate gradient learning in spiking neural networks*, 2019. arXiv: 1901.09948 [cs.NE].

[21] T. A. d. Oliveira, M. P. d. Silva, E. H. B. Maia, A. M. d. Silva, and A. G. Taranto, "Virtual Screening Algorithms in Drug Discovery: A Review Focused on Machine and Deep Learning Methods," *Drugs and Drug Candidates*, vol. 2, no. 2, pp. 311–334, Jun. 2023, ISSN: 2813-2998. DOI: 10.3390/ddc2020017. [Online]. Available: https://www.mdpi.com/2813-2998/2/2/17 (visited on 2024-03-06).

[22] R. Qureshi *et al.*, "AI in drug discovery and its clinical relevance," *Heliyon*, vol. 9, no. 7, e17575, Jun. 2023, ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2023.e17575. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10302550/ (visited on 2023-09-26).

[23] R. V. Devi, S. S. Sathya, and M. S. Coumar, "Evolutionary algorithms for de novo drug design – a survey," *Applied Soft Computing*, vol. 27, pp. 543–552, 2015, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2014.09.042. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494614004955.

[24] D. D. Martinelli, "Generative machine learning for de novo drug discovery: A systematic review," *Computers in Biology and Medicine*, vol. 145, p. 105 403, 2022, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105403. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482522001950.

[25] N. Arul Murugan, G. Ruba Priya, G. Narahari Sastry, and S. Markidis, "Artificial intelligence in virtual screening: Models versus experiments," *Drug Discovery Today*, vol. 27, no. 7, pp. 1913–1923, Jul. 2022, ISSN: 1359-6446. DOI: 10.1016/j.drudis.2022.05.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359644622002033 (visited on 2023-12-13).

[26] N. Singh, P. Vayer, S. Tanwar, J.-L. Poyet, K. Tsaioun, and B. O. Villoutreix, "Drug discovery and development: Introduction to the general public and patient groups," *Frontiers in Drug Discovery*, vol. 3, 2023, ISSN: 2674-0338. DOI: 10.3389/fddsv.2023.1201419. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fddsv.2023.1201419.

[27] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988. DOI: 10.1021/ci00057a005. eprint: https://doi.org/10.1021/ci00057a005. [Online]. Available: https://doi.org/10.1021/ci00057a005.

[28] D. Boldini, D. Ballabio, V. Consonni, R. Todeschini, F. Grisoni, and S. A. Sieber, "Effectiveness of molecular fingerprints for exploring the chemical space of natural products," *Journal of Cheminformatics*, vol. 16, no. 1, p. 35, Mar. 2024, ISSN: 1758-2946. DOI: 10.1186/s13321-024-00830-3. [Online]. Available: https://doi.org/10.1186/s13321-024-00830-3 (visited on 2024-06-02).

[29] H. L. Morgan, "The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service.," *Journal of Chemical Documentation*, vol. 5, no. 2, pp. 107–113, May 1965, ISSN: 0021-9576, 1541-5732. DOI: 10.1021/c160017a018. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/c160017a018 (visited on 2023-12-07).

[30] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of mdl keys for use in drug discovery," *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 6, pp. 1273–1280, 2002, PMID: 12444722. DOI: 10.1021/ci010132r. eprint: https://doi.org/10.1021/ci010132r. [Online]. Available: https://doi.org/10.1021/ci010132r.

[31] T. E. Hadfield, J. Scantlebury, and C. M. Deane, "Exploring the ability of machine learning-based virtual screening models to identify the functional groups responsible for binding," *Journal of Cheminformatics*, vol. 15, no. 1, p. 84, Sep. 2023, ISSN: 1758-2946. DOI: 10.1186/s13321-023-00755-3. [Online]. Available: https://doi.org/10.1186/s13321-023-00755-3 (visited on 2024-06-02).

[32] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "Guacamol: Benchmarking models for de novo molecular design," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1096–1108, 2019, PMID: 30887799. DOI: 10.1021/acs.jcim.8b00839. eprint: https://doi.org/10.1021/acs.jcim.8b00839. [Online]. Available: https://doi.org/10.1021/acs.jcim.8b00839.

[33] M. A. Skinnider, R. G. Stacey, D. S. Wishart, and L. J. Foster, *Deep Generative Models Enable Navigation in Sparsely Populated Chemical Space*, Jan. 2021. DOI: 10.26434/chemrxiv.13638347.v1. [Online]. Available: https://chemrxiv.org/engage/chemrxiv/article-details/60c75449bb8c1ae12a3dc1d2 (visited on 2024-05-26).

[34] A. Vaswani *et al.*, *Attention is all you need*, 2023. DOI: https://doi.org/10.48550/arXiv.1706.03762. arXiv: 1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762.

[35] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[36] R. Özçelik, S. De Ruiter, and F. Grisoni, "Structured State-Space Sequence Models for De Novo Drug Design," Chemistry, preprint, Sep. 2023. DOI: 10.26434/chemrxiv-2023-jwmf3. [Online]. Available: https://chemrxiv.org/engage/chemrxiv/article-details/65168004ade1178b24567cd3 (visited on 2024-03-07).

[37] Y. Wang, H. Zhao, S. Sciabola, and W. Wang, "Cmolgpt: A conditional generative pre-trained transformer for target-specific de novo molecular generation," *Molecules*, vol. 28, no. 11, 2023, ISSN: 1420-3049. DOI: 10.3390/molecules28114430. [Online]. Available: https://www.mdpi.com/1420-3049/28/11/4430.

[38] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990. DOI: 10.1109/5.58356.

[39] L. Smirnova *et al.*, "Organoid intelligence (oi): The new frontier in biocomputing and intelligence-in-a-dish," *Frontiers in Science*, vol. 1, p. 1 017 235, Feb. 2023. DOI: 10.3389/fsci.2023.1017235.

[40] J. K. Eshraghian *et al.*, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023. DOI: 10.1109/JPROC.2023.3308088.

[41] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers in Neuroscience*, vol. 15, 2021, ISSN: 1662-453X. DOI: 10.3389/fnins.2021.638474. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.638474.

[42] D. L. Manna, A. Vicente-Sola, P. Kirkland, T. J. Bihl, and G. D. Caterina, *Frameworks for snns: A review of data science-oriented software and an expansion of spyketorch*, 2023. arXiv: 2302.07624 [cs.SE].

[43] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, *Incorporating learnable membrane time constant to enhance learning of spiking neural networks*, 2021. arXiv: 2007.05785 [cs.NE].

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0. [Online]. Available: https://www.nature.com/articles/323533a0 (visited on 2024-05-29).

[45] S. Gunasekaran, J. Eshraghian, R. Zhu, and Z. Kuncic, "Knowledge distillation through time for future event prediction," in *The Second Tiny Papers Track at ICLR 2024*, 2024. [Online]. Available: https://openreview.net/forum?id=JBSMl0nAFa.

[46] S. B. Shrestha and G. Orchard, "SLAYER: spike layer error reassignment in time," *CoRR*, vol. abs/1810.08646, 2018. arXiv: 1810.08646. [Online]. Available: http://arxiv.org/abs/1810.08646.

[47] B. Peng *et al.*, *Rwkv: Reinventing rnns for the transformer era*, 2023. arXiv: 2305.13048 [cs.CL].

[48] X. Wang *et al.*, *State space model for new-generation network alternative to transformers: A survey*, 2024. arXiv: 2404.09516 [cs.LG].

[49] M. Grootendorst, *A Visual Guide to Mamba and State Space Models*. [Online]. Available: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state (visited on 2024-05-31).

[50] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018. DOI: 10.1109/MM.2018.112130359.

[51] D. Van Tilborg, A. Alenicheva, and F. Grisoni, "Exposing the Limitations of Molecular Machine Learning with Activity Cliffs," *Journal of Chemical Information and Modeling*, vol. 62, no. 23, pp. 5938–5951, Dec. 2022, ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.2c01073.

[Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jcim.2c01073 (visited on 2023-12-04).

[52] T. M. Keck, W. S. John, P. W. Czoty, M. A. Nader, and A. H. Newman, "Identifying Medication Targets for Psychostimulant Addiction: Unraveling the Dopamine D3 Receptor Hypothesis," *Journal of Medicinal Chemistry*, vol. 58, no. 14, pp. 5361–5380, Jul. 2015, ISSN: 0022-2623, 1520-4804. DOI: 10.1021/jm501512b. [Online]. Available: https://pubs.acs.org/doi/10.1021/jm501512b (visited on 2024-06-24).

[53] A. Gaulton *et al.*, "The ChEMBL database in 2017," *Nucleic Acids Research*, vol. 45, no. D1, pp. D945–D954, Jan. 2017, ISSN: 1362-4962. DOI: 10.1093/nar/gkw1074.

[54] M. Nascimben and L. Rimondini, "Molecular Toxicity Virtual Screening Applying a Quantized Computational SNN-Based Framework," *Molecules*, vol. 28, no. 3, p. 1342, Jan. 2023, ISSN: 1420-3049. DOI: 10.3390/molecules28031342. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9919191/ (visited on 2024-03-07).

[55] S. Riniker and G. A. Landrum, "Open-source platform to benchmark fingerprints for ligand-based virtual screening," *Journal of Cheminformatics*, vol. 5, no. 1, p. 26, May 2013, ISSN: 1758-2946. DOI: 10.1186/1758-2946-5-26. [Online]. Available: https://doi.org/10.1186/1758-2946-5-26 (visited on 2023-12-07).

[56] D. Rogers and M. Hahn, "Extended-Connectivity Fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, May 2010, ISSN: 1549-9596, 1549-960X. DOI: 10.1021/ci100050t. [Online]. Available: https://pubs.acs.org/doi/10.1021/ci100050t (visited on 2023-12-07).

[57] W. A. Warr, M. C. Nicklaus, C. A. Nicolaou, and M. Rarey, "Exploration of ultralarge compound collections for drug discovery," *Journal of Chemical Information and Modeling*, vol. 62, no. 9, pp. 2021–2034, 2022, PMID: 35421301. DOI: 10.1021/acs.jcim.2c00224. eprint: https://doi.org/10.1021/acs.jcim.2c00224. [Online]. Available: https://doi.org/10.1021/acs.jcim.2c00224.

[58] *Intel Builds World's Largest Neuromorphic System to Enable More Sustainable AI*, publisher: Intel Public Relations Team, Apr. 2024. [Online]. Available: https://www.intel.com/content/www/us/en/newsroom/news/intel-builds-worlds-largest-neuromorphic-system.html.

[59] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, *Optuna: A next-generation hyperparameter optimization framework*, 2019. arXiv: 1907.10902 [cs.LG].

[60] J. You, J.-W. Chung, and M. Chowdhury, "Zeus: Understanding and optimizing GPU energy consumption of DNN training," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, Boston, MA: USENIX Association, Apr. 2023, pp. 119–139, ISBN: 978-1-939133-33-5. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/you.

[61] C. Lv, J. Xu, and X. Zheng, "Spiking Convolutional Neural Networks for Text Classification," Sep. 2022. [Online]. Available: https://openreview.net/forum?id=pgU3k7QXuz0 (visited on 2023-12-06).

[62] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv:1301.3781 [cs], Sep. 2013. DOI: 10.48550/arXiv.1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781 (visited on 2023-12-06).

[63] A. Rao, P. Plank, A. Wild, and W. Maass, "A Long Short-Term Memory for AI Applications in Spike-based Neuromorphic Hardware," *Nature Machine Intelligence*, vol. 4, no. 5, pp. 467–479, May 2022, ISSN: 2522-5839. DOI: 10.1038/s42256-022-00480-w. [Online]. Available: https://www.nature.com/articles/s42256-022-00480-w (visited on 2023-12-06).

[64] D. Polykovskiy *et al.*, *Molecular sets (moses): A benchmarking platform for molecular generation models*, 2020. arXiv: 1811.12823 [cs.LG].

[65] M. Szafarczyk, P. Ludynia, and P. Kukla, "A python library for efficient computation of molecular fingerprints," 2024. DOI: https://doi.org/10.48550/arXiv.2403.19718. arXiv: 2403.19718. [Online]. Available: https://arxiv.org/abs/2403.19718.

[66] V.-K. Tran-Nguyen, C. Jacquemard, and D. Rognan, "Lit-pcba: An unbiased data set for machine learning and virtual screening," *Journal of Chemical Information and Modeling*, vol. 60, no. 9, pp. 4263–4273, 2020, PMID: 32282202. DOI: 10.1021/acs.jcim.0c00155. eprint: https://doi.org/10.1021/acs.jcim.0c00155. [Online]. Available: https://doi.org/10.1021/acs.jcim.0c00155.

[67] N. J. Szymanski *et al.*, "An autonomous laboratory for the accelerated synthesis of novel materials," *Nature*, vol. 624, no. 7990, pp. 86–91, Dec. 2023, ISSN: 1476-4687. DOI: 10.1038/s41586-023-06734-w. [Online]. Available: https://www.nature.com/articles/s41586-023-06734-w (visited on 2024-05-26).

[68] S. Luo, W. Gao, Z. Wu, J. Peng, C. W. Coley, and J. Ma, *Projecting molecules into synthesizable chemical spaces*, 2024. arXiv: 2406.04628 [cs.CE]. [Online]. Available: https://arxiv.org/abs/2406.04628.

[69] C. Li, Z. Yao, X. Wu, M. Zhang, and Y. He, "Deepspeed data efficiency: Improving deep learning model quality and training efficiency via efficient data sampling and routing," Dec. 2022. [Online]. Available: https://www.microsoft.com/en-us/research/publication/deepspeed-data-efficiency-improving-deep-learning-model-quality-and-training-efficiency-via-efficient-data-sampling-and-routing/.

[70] F. Ortiz *et al.*, *Energy-efficient on-board radio resource management for satellite communications via neuromorphic computing*, 2023. arXiv: 2308.11152 [eess.SY]. [Online]. Available: https://arxiv.org/abs/2308.11152.

[71] Y.-C. Chen, "Beware of docking!" *Trends in Pharmacological Sciences*, vol. 36, no. 2, pp. 78–95, 2015, ISSN: 0165-6147. DOI: https://doi.org/10.1016/j.tips.2014.12.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165614714002144.

[72] M. Xu, C. Shen, J. Yang, Q. Wang, and N. Huang, "Systematic Investigation of Docking Failures in Large-Scale structure-Based Virtual Screening," *ACS Omega*, vol. 7, no. 43, pp. 39 417–39 428, Oct. 2022, ISSN: 2470-1343. DOI: 10.1021/acsomega.2c05826. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9632257/ (visited on 2024-06-26).

[73] M. J. Rasch *et al.*, "A flexible and fast pytorch toolkit for simulating training and inference on analog crossbar arrays," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4. DOI: 10.1109/AICAS51828.2021.9458494.

[74] S.-i. Yi, J. D. Kendall, R. S. Williams, and S. Kumar, "Activity-difference training of deep neural networks using memristor crossbars," *Nature Electronics*, Nov. 2022, ISSN: 2520-1131. DOI: 10.1038/s41928-022-00869-w. [Online]. Available: https://www.nature.com/articles/s41928-022-00869-w (visited on 2024-01-22).

[75] G. Shen *et al.*, *Astrocyte-enabled advancements in spiking neural networks for large language modeling*, 2023. arXiv: 2312.07625 [cs.NE].

[76] J. H. Lee, S. Haghighatshoar, and A. Karbasi, *Exact gradient computation for spiking neural networks through forward propagation*, 2023. arXiv: 2210.15415 [cs.NE].

[77] M. Xiao, Q. Meng, Z. Zhang, D. He, and Z. Lin, *Online training through time for spiking neural networks*, 2022. arXiv: 2210.04195 [cs.NE].

[78] neureka, *NEUREKA – A smart, neural-silico-computo device for drug discovery*. [Online]. Available: http://neureka.gr/.

[79] N. Yin, M. Wang, Z. Chen, G. D. Masi, B. Gu, and H. Xiong, *Dynamic spiking graph neural networks*, 2023. arXiv: 2401.05373 [cs.NE].

[80] S. Snyder *et al.*, "Transductive spiking graph neural networks for loihi," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24, ¡conf-loc¿, ¡city¿Clearwater¡/city¿, ¡state¿FL¡/state¿, ¡country¿USA¡/country¿, ¡/conf-loc¿: Association for Computing Machinery, 2024, pp. 608–613, ISBN: 9798400706059. DOI: 10.1145/3649476.3660366. [Online]. Available: https://doi.org/10.1145/3649476.3660366.

[81] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar, *Chemberta-2: Towards chemical foundation models*, 2022. arXiv: 2209.01712 [cs.LG].

[82] Y. Wang, Y. Zeng, J. Tang, and B. Xu, "Biological Neuron Coding Inspired Binary Word Embeddings," *Cognitive Computation*, vol. 11, no. 5, pp. 676–684, Oct. 2019, ISSN: 1866-9956, 1866-9964. DOI: 10.1007/s12559-019-09643-1. [Online]. Available: http://link.springer.com/10.1007/s12559-019-09643-1 (visited on 2024-04-29).

[83] Y. Liang *et al.*, *Can a fruit fly learn word embeddings?* 2021. arXiv: 2101.06887 [cs.CL].

[84] J. Arús-Pous, T. Blaschke, S. Ulander, J.-L. Reymond, H. Chen, and O. Engkvist, "Exploring the GDB-13 chemical space using deep generative models," *Journal of Cheminformatics*, vol. 11, no. 1, p. 20, Mar. 2019, ISSN: 1758-2946. DOI: 10.1186/s13321-019-0341-z. [Online]. Available: https://doi.org/10.1186/s13321-019-0341-z (visited on 2024-06-06).

Appendices

A Additional analysis

Table 8.1: Balanced accuracy, recall, specificity and AUC score of the best ANN and SNN after hyperparameter search for different time steps.

Fingerprint	Size	Time steps	Model	Balanced accuracy	Recall	Specificity	AUC
MACCS keys	167	32	SNN	77.3 %	0.748	0.798	0.85
MACCS keys	167	21	SNN	77.3%	0.891	0.654	0.86
MACCS keys	167	16	SNN	75.7 %	0.865	0.650	0.82
MACCS keys	167	8	SNN	78.6 %	0.849	0.724	0.83
MACCS keys	167	4	SNN	50 %	0	1	0.5
MACCS keys	167	2	SNN	50 %	0	1	0.5
MACCS keys	167	1	ANN	78.4 %	0.873	0.695	0.87
Morgan Fingerprint	1024	32	SNN	81.2 %	0.822	0.802	0.87
Morgan Fingerprint	1024	21	SNN	80.4 %	0.894	0.716	0.85
Morgan Fingerprint	1024	16	SNN	80.0 %	0.822	0.777	0.85
Morgan Fingerprint	1024	8	SNN	80.4 &	0.893	0.716	0.87
Morgan Fingerprint	1024	4	SNN	50 %	0	1	0.5
Morgan Fingerprint	1024	2	SNN	50 %	0	1	0.5
Morgan Fingerprint	1024	1	ANN	81.7%	0.897	0.736	0.91