Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

A Comparison of Selective State Space Models and Transformers for Single-Step Retrosynthetic Reaction Prediction

vorgelegt von

Lennart Roth

MIN-Fakultät

Fachbereich Informatik

Studiengang: Informatik

Matrikelnummer: 7495866

Abgabedatum: 24.09.2024

Erstgutachter: Dr. Steffen Remus

Zweitgutachterin: Dr. Katrin Schöning-Stierand

Abstract

This thesis compares Selective State Space Models, specifically Mamba (Gu and Dao, 2023), with Chemformer (R Irwin et al., 2022), a Transformer model implementation, on the task of single-step retrosynthetic reaction prediction. The aim is to evaluate the scalability of both models in terms of model size, inference time, and performance. Emphasis is placed not only on comparing both models in terms of quantitative metrics such as molecular and token-based accuracy but also on a task-tailored qualitative evaluation of sequential and chemical features of the generated compounds.

The Transformer architecture is highly parallelizable and thus efficient and scalable regarding dataset size. This comes at the cost of slow inference time and high computational complexity. In contrast, the State Space architecture scales linearly in inference time but lacks parallelizability. The Mamba model strikes a balance between the two characteristics and places emphasis on handling long-range dependencies and input selectivity. This was demonstrated by applying Mamba to a DNA sequence classification task with long context and sparse input (Gu and Dao, 2023). I show that Mamba is a well-suited candidate for sequence prediction tasks such as SMILES-based single-step retrosynthetic reaction prediction, as inference time is a precious commodity in the high-throughput generation of new molecules in the chemical space.

My experiments confirm this and show that Mamba is orders of magnitude faster (379%) for inference and on par with the Transformer in training time with only a tiny loss of accuracy. This shows that the Mamba model can be used in chemical tasks and masked pre-training can be used equivalently to existing training routines for Mamba models. Further evaluations demonstrate that an ensemble model combining both architectures gives promising results that are better than either model alone. This thesis also gives a trivial proposal for implementing the Mamba model with other SOTA architectures incorporating additional chemical information, which has great potential for further improvement. This is supported by experiments using a mixture model of alternating Mamba and Attention blocks outperforming both stand-alone models (Dao and Gu, 2024).

Contents

List of Figures

1 Introduction

1.1 Relevance

The process of discovering new molecules with targeted properties is one of the most important tasks in organic chemistry. This technology has a wide and diverse range of economic, social and scientific applications. It is not only the backbone for the design of new drugs, but also for various tasks in materials science and agricultural research. Because of this wide range of applications, the design of chemical compounds is an active area of research in chemistry and, increasingly, in computer science. The combination of these two fields is commonly known as cheminformatics and aims to apply the principles and techniques of data science and computer science to chemical data. This data can be, for example, a large database of "chemical molecules, structures, properties, spectra and activities" (Wishart, 2007p.1).

1.2 Traditional Cheminformatics

One of the most significant tasks in this field is the design of new drugs. In particular, the problem of designing a new molecule with specific chemical or biological properties remains a challenging one. This is mainly due to the large size and complexity of the chemical space of possible molecules. It is generally estimated to contain about 10^{60} drug-like molecules (Lipinski et al., 1997; Drew et al., 2012; Mullard et al., 2017). Bohacek et al. (1996) estimate this size to be drastically lower. The space of molecules up to 30 atoms has even been estimated to be $10^{20} - 10^{24}$ by Ertl (2003). Although these numbers are much smaller than the previous estimate of 10^{60}, they are still too large for a complete combinatorial exploration of chemical space.

Traditional drug design methods rely heavily on manual labor. This makes the drug design process expensive, time-consuming and error-prone, with the development of a new drug taking between 7 and 10 years and costing up to US$ 4.54 billion (Schlander et al., 2021). Some of these problems can be mitigated using computer-aided drug design systems. These systems often involve combinatorial chemistry approaches that aim to generate an large number of possible compounds at high throughput. While the

accuracy and consistency of such approaches can be very good, the huge size of chemical space makes it impossible to represent using combinatorial approaches. This means that these approaches cannot be used to represent the entire chemical space because they often rely on existing databases, which themselves represent only a small fraction of the number of possible molecules. One way of making use of such huge but far from complete databases is the technique of virtual screening (Walters et al., 1998). This can be used to reduce the size of a virtual library of compounds to a more manageable size. A general workflow of such a filtering through virtual screening is shown in Figure 1.1. Problems with traditional ligand and also structure based virtual screening are limited accuracy, the generation of many false positives and low potency compounds (Stumpfe and Bajorath, 2020). This is the case because, traditional virtual screening uses mainly substructure, pharmacophore or topological fingerprint methods to filter possible molecules (Gimeno et al., 2019). These problems are caused by statistical and combinatorial properties, which have been known for years (Stumpfe and Bajorath, 2020). This is why the field of computer-aided drug design has not seen any substantial developments for most of the 90s, 00s and 10s.

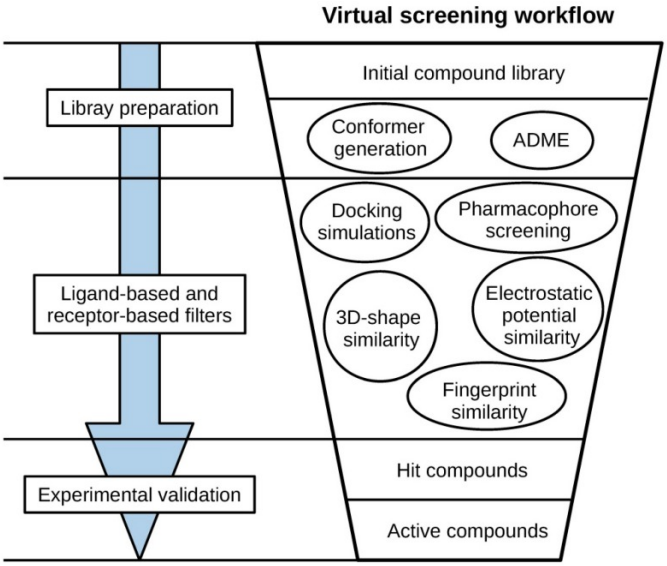


Figure 1.1: General scheme of a virtual screening workflow. [1]

1.3 Machine Learning for De Novo Drug Desing

However, the field of cheminformatics has received much more attention in recent years due to the huge progress in machine learning, which has led to applications of artificial Intelligence (AI) in many different fields. As a result, many computational approaches have been proposed for many tasks. One of them is the de novo generation of drug-like molecules with desired properties (Schneider and Schneider, 2016; Blaschke et al., 2020). This is one step of the complex process of developing new medicine. De novo means creating molecules from scratch. This is in contrast to methods that try to optimize existing molecules. While virtual screening relies on existing databases, de novo design

1. Figure taken from Gimeno et al. (2019).

generates structures without a starting point. Because of these differences, virtual screening focuses on a small part of the available chemical space, which is still a large number of processed molecules, whereas de novo design could theoretically cover the entire space of possible molecules. Recent advances in de novo drug design, thanks in part to advances in machine learning, show promising results in generating chemically valid molecular structures and also molecules with desired properties (Blaschke et al., 2020). This shows that such models can successfully navigate the chemical space of molecules. However, one problem with these approaches is the synthesizability of the generated molecules. While they can produce accurate virtual molecules, the generation process does not use any information about how a target molecule could be synthesized. As a result, the often promising results are not very useful for real pharmaceutical use.

To tackle this problem, some of the more recent de novo design approaches use techniques such as scaffold hopping to combine two known scaffolds in a reaction to create a new molecule (Guo et al., 2023). In scaffold decoration, an existing base compound is decorated with functional groups to form a new molecule (Fialková et al., 2021). Another approach is to iteratively build molecules, not from their atoms, but by using already known and commonly occurring motifs. These motifs can be ring structures or chemical groups that are common to many molecules (Jin et al., 2020). Figure 1.2 shows the different approaches discussed above from Loeffler et al. (2024). While these technologies can increase the synthesizability and chemical validity of the generated molecules, the generation process is often more complex than the simple combination of motifs or scaffolds. This results in generated molecules that appear directly synthesisable to the untrained eye, but in reality may not be as easily synthesisable as the model predicted, or worse, may not even be chemically stable. The reason for this discrepancy is that a huge amount of chemical information is missing from such algorithms. Most of them use the 2D structure of the molecules for generation. However, the properties of such molecules depend on, e.g., their 3D orientation and many other chemical properties (Venkatasubramanian and Mann, 2022).

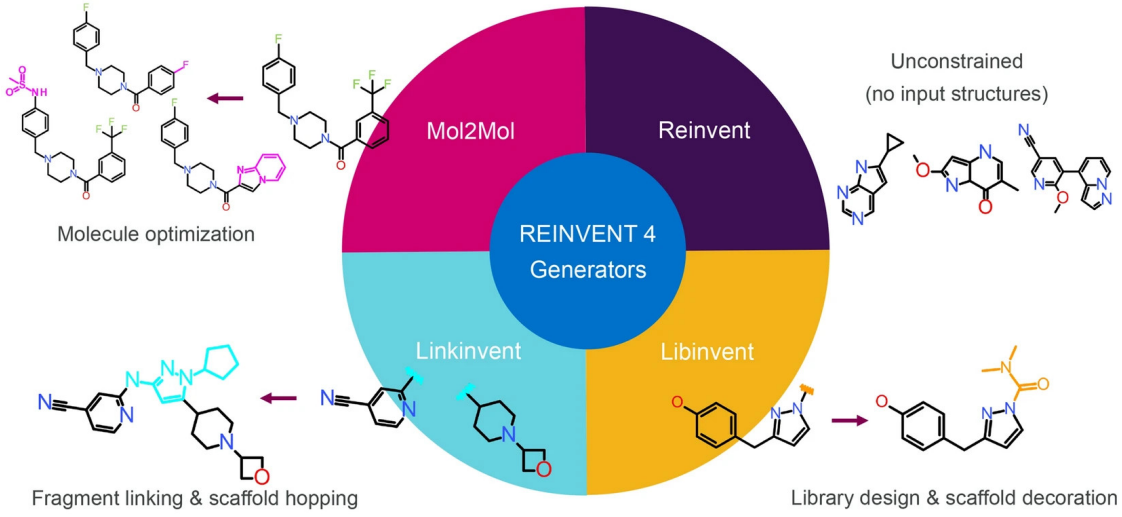


Figure 1.2: Four types of molecular de novo generators. [2]

2. Figure taken from Loeffler et al. (2024).

1.4 Motivation of this Thesis

A particular emphasis will be placed on the **efficiency, training and inference cost** of any machine learning model evaluated. This is of great importance especially for any task in the chemical domain, since it has to handle a vastly larger space of possible generated sequences (molecules described in the SMILES language) while the number of potential elements in the vocabulary is relatively limited.

The Transformer, first introduced by Vaswani (2017), gained popularity in many Natural Language Processing (NLP) tasks as it allowed for higher performance, scalability and attention aware to the context of the input sequence. The key advantages the Transformer brought to the domain of NLP include parallelization during training and the use of the attention mechanism. The parallelization lead to a good scalability both in model dimension and amount of training data. The use of the attention mechanism allows to weigh the importance of different tokens dynamically on the input, which results in a more nuanced understanding of the input data. This also lead to the ability to capture long range dependencies of single tokens in the input sequence improving tasks such as selective copying but also more general ones such as reasoning over a longer context length.

These advantages where also found to be helpful for tasks in computer vision, which resulted in transformer based models directly dedicated to, e.g., image classification (Alexey, 2020). Also in the signal processing domain tasks such as speech enhancement in audio recordings or synthetic voice generation were improved by Transformers (Li et al., 2019; Wang et al., 2021). Cheminformatics, the field of this thesis is also experiencing advances through the development of the transformer. This is demonstrated by the Chemformer, which is close to the state of the art in this field.

All the described use-cases show how versatile the architecture is, however it also has some downsides. The most predominant one being the high computational complexity. Since the self attention mechanism computes attention scores for each token with every other token in the sequence it has a quadratic complexity with respect to the input sequence. This leads to a limited context size and relatively slow inference.

With the effort to reduce the long inference times of transformers and simultaneously the need for fast parallelizable training of large language models, one of the advances, called Selective State Space Model (SSSM) (Gu and Dao, 2023), is using the principle of hidden Markov chains in combination with a recurrent and a convolutional representation to improve the long context length and selectivity of information in the input together with the possibility of parallel training strategies and fast inference. Especially the fast inference time which is linear in regard to the input length is crucial for the application of large language models (LLMs) for example in real time scenarios or cheminformatics, the application of this thesis. This is archieved through the multiple representations the model can be defined by, which can be interchanged even after the training is completed. More information on this is given in Chapter 2.

While chemical screening methods have to scan all possibilities of the chemical domain to find new molecule candidates, every other approach of multistep reaction prediction is also constrained by the inference time of its underlying model used for prediction of a single step reaction. The use of a vastly faster and optimizable inference model would allow for faster exploring of the chemical space. In this thesis I will therefore explore the use cases of Selective State Space Models in the chemical reaction

prediction domain and compare its performance and speed against a SOTA Transformer architecture such as the Chemformer (R Irwin et al., 2022).

1.5 Hypothesis

While SOTA transformer architectures are among the best performing approaches with scale for the task of single step reaction prediction, SSSMs are expected to produce comparable results at same scale. This hypothesis is supported by Gu and Dao (2023) who showed that Mamba can archive comparable or better results on many tasks that require understanding of long range dependencies. Additionally, one of the tasks Gu and Dao (2023) evaluated Mamba on, is the DNA classification. This task is inherently similar to reaction prediction since the used representations of DNA and molecules are similar in character distribution and context length. In this thesis we focus on investigating the training and inference, efficiency and performance of SSSMs compared to Transformer-based approaches and additionaly test different methods of combining both for the given task.

We expect and show that SSSMs almost completely match the performance of transformers in respect to the model size while being equally parallelizable in training and providing linear inference time on the input sequence length.

1.6 Structure of this Thesis

Chapter 1 gives a brief introduction to cheminformatics and provides a hypothesis and motivation for this thesis. Chapter 2 gives a brief introduction into the SMILES language, reaction prediction tasks, the transformer architecture and recalls the most important properties of them. It is followed by a comprehensive explanation of the SSSM, in particular the Mamba architecture. If the reader is familiar with these topics, this chapter can be skipped. In Chapter 3 the different approaches to single step reaction prediction are discussed. While traditional enumerative approaches such as virtual screening are described in the introduction, this chapter focuses on more recent AI-based approaches to retrosynthetic reaction prediction. In particular, the SOTA architecture NAG2G is explained, which will later be used to integrate the Mamba architecture. The next Chapter (Chapter 4), lists details of the datasets and architectures used and also explains the procedures used to evaluate all the models in terms of performance, scalability and inference time. The evaluation of the models described in Chapter 4 is finally performed in Chapter 5, where the performance of the two models is compared, firstly in terms of the model metrics and secondly in terms of the generated SMILES output chemical and linguistic features. Other figures used in this chapter can be found in Appendix A.

2 Background

2.1 Reaction Prediction

The problem of generating molecules that have the needed properties but are difficult to synthesize can in part be addressed by following the traditional manual approach of finding a sequence of reactions that lead to the desired molecule. These approaches can be broadly divided into two groups. The *retrosynthetic* models work from a given virtual generated molecule in multiple steps to its precursors in so-called reverse reactions until all precursors are known and available (Hasic and Ishida, 2021). This is done by training a generative model on known discovered reactions, often from industry patents. This model can then predict the precursors of a molecule and the type of reaction that would be used to go from the precursors to the target molecule. The opposite of this idea is called *forward synthesis*. Here the input is one or more reactants and the model generates the product that would result from that reaction (Tu et al., 2023). The result of both approaches is called a synthetic route which could be therefore either retro-synthetically or forward-synthetically constructed. Figure 2.1 shows such a synthetic route for a target molecule. At each node of the tree there are one or more molecules, concatenated by a plus sing. Each incoming edge to a node represents a possible reaction to create the molecules stated at this node. Additional solvents that are needed for this reaction can be given next to the edge. Otherwise, the reaction uses the molecules from the node from which the edge originates. Therefore, the synthetic route in Figure 2.1 shows two possible routes to create the target molecule (Świst et al., 2005).

The reaction prediction can also be done using an expert system based on hardcoded rules. This is called template based retrosynthesis. In contrast, the template-free approach interprets a reaction as a translation task from a textual representation of the input reactant molecules to a language description of a product molecule. The advantages of the template-based approach are fast computation times and accurate, rule-based results. The low computational effort allows this molecule generation to be parallelized at a very high level (Patel et al., 2020). However, this is not scalable regarding the estimated size of the chemical space, especially for the exploration of multi-reactant or multistep reactions. The machine learning approach, however, combines the

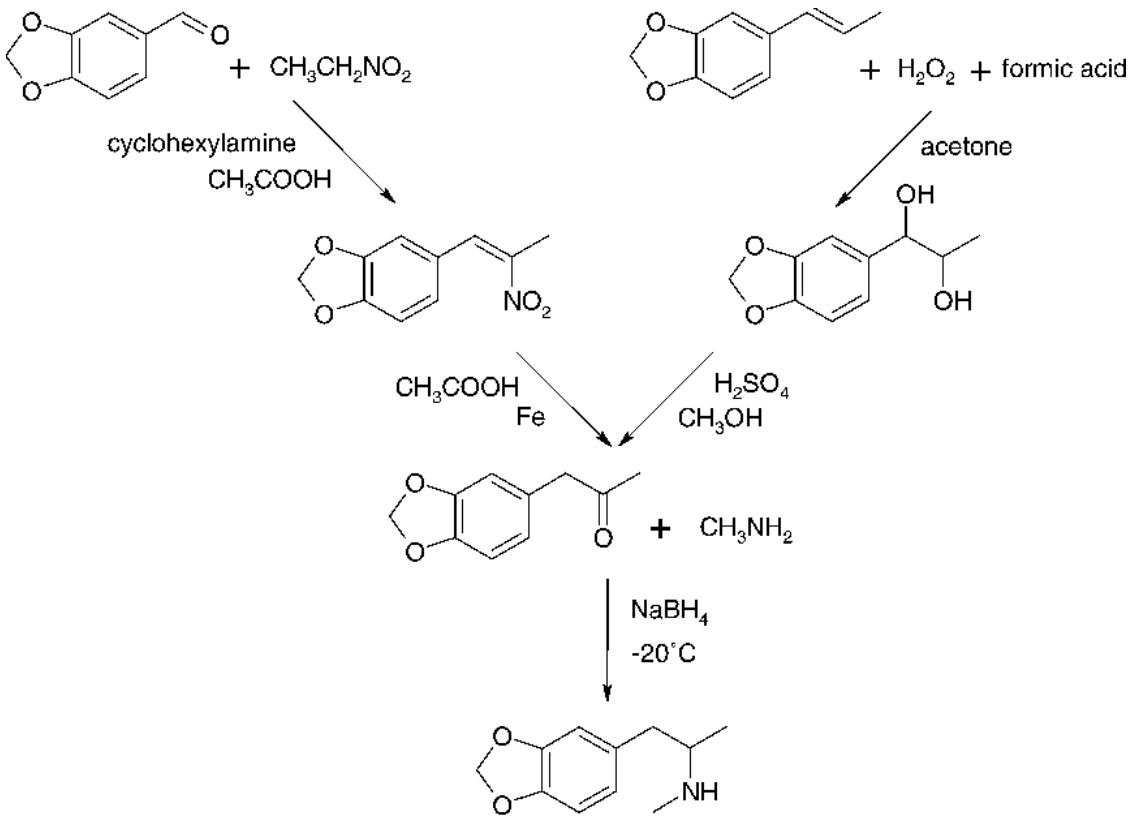


Figure 2.1: Example of two synthetic routes for the molecule 1-(3,4-methylenedioxyphenyl)-2-propanone. As a retrosynthetic route one would start at the finished target molecules and begin to construct this tree from bottom to top. And from top to bottom for the forward synthetic route. [1]

advantage of interpretability through reaction steps with the advantages of machine learning, which otherwise lacks the interpretability aspect. Additional advantages of the template-free approach are generally more diverse molecules than the product. In state-of-the-art models, these two approaches archieve similar accuracies on popular metrics. This shows that template-free models are indeed able to learn the chemical rules and language to model molecules. This is further demonstrated by the high validity scores of the generated molecules (Yao et al., 2024).

Another promise of machine learning approaches is that they can learn deep chemical relationships between atoms and therefore predict new reactions, whereas the template-based approach can only use existing reactions and apply them to molecules not used in that reaction.

As Hassen et al. (2022) show, the importance of single-step synthesis prediction has a large influence on the performance of multi-step reaction prediction. As a result, much research has focused on the task of single step prediction in both retrosynthetic and forward synthetic ways. Following this idea, this work will further explore the task of reaction prediction to generate synthesizable molecules using machine learning.

1. Figure taken from Świst et al. (2005).

2.2 SMILES

The Simplified Molecular Input Line Entry System (SMILES) language is one of the most commonly used notations for describing molecules in chemistry (Weininger, 1988). Because of its very compact format and machine-readable syntax, it is often used in chemical databases to search or represent molecules. As SMILES is a context-free grammar, it is also widely used in cheminformatics as input to language models. An example is given in 2.2. However, SMILES has some drawbacks. Firstly, it is not a unique representation of a molecule. A molecule can have several SMILES representing it. This can be a big problem for any language model learning the syntax of SMILES. In addition, SMILES do not encode any spatial information about the atoms in the molecule. This can be very important for models that predict reactions or chemical properties of molecules, as they can have a complicated 3D structure where atoms that are not connected by a bond still interact with other atoms in the vicinity. This is called stereochemistry and can also define relative orientations between bonds and atoms. However, it is not possible to define stereochemistry in the basic SMILES system. Other formats, particularly graph-based representations, use coordinates for atoms in 2D or 3D to precisely define the structure of the molecule (Jin et al., 2018). The sequential generation of strings does not follow the defined SMILES syntax. A generated molecule may therefore be syntactically or chemically invalid. This can be a problem when training a model, since a bias towards molecules that can be generated easily and correctly could be induced (Schoenmaker et al., 2023). The strict syntax of SMILES also makes it very sensitive to changes in the string. For example, the order of the atoms or the choice of bond types can lead to very different molecules.

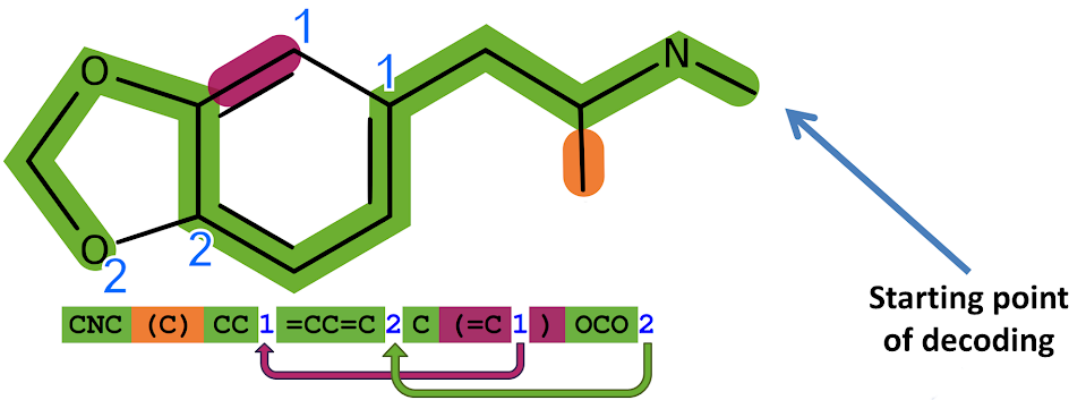


Figure 2.2: The SMILES molecule description language. [2]

2.3 Processing SMILES

To process a SMILES reaction it has to be converted to IDs. This is done by first tokenizing the reaction string and then returning the indices of the these tokens in the vocabulary of the tokenizer. In the case of the work of R Irwin et al. (2022) the tokenizer splits the string into tokens that resemble single molecules such as "C" or "N". In addition, it takes into consideration the differences between capital and non-capital letters and

2. Figure taken from Krenn et al. (2020).

the rest of the SMILES syntax such as "(" or "=". Very common functional groups and other common chemical building blocks are also included in the vocabulary such as ring structures. A visualization of the SMILES tokenizer can be seen in Figure 2.3.

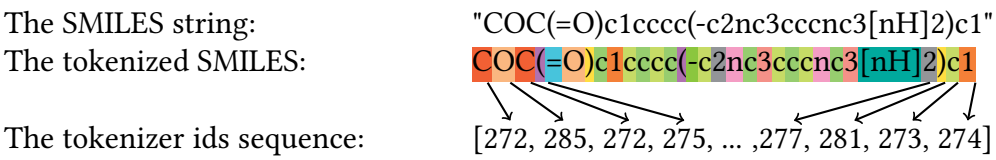


Figure 2.3: An example SMILES, as a string (top), tokenized (middle), in ids format (bottom). The colors represent the separate tokens where the same token has the same color. Except for the "c" token where a alternating green is chosen to show that the string "cccc" is converted into 4 separate tokens.

2.4 Probabilistic Machine Learning

This section is based on (Murphy, 2023, Chapter 1 and Chapter 5, Section 29).
The string we want to generate is the SMILES string that represents the product in a reaction. The prompt to the LLM is therefore the SMILES string of the reactants. If the prompt or the generated output has to contain multiple molecules this is done by describing them in a single concatenated string in SMILES format. For each training we then concatenate these two SMILES strings, the input and the target, together. The model is then trained on many examples of these SMILES reactions. For each reaction the model is given the first n elements of the reaction and try to predict one in an autoregressive fashion. This is done for all n between 1 and the length of the whole sequence. This results in a single string in the data being used for many training examples with context length from 1 to the length of the whole sequence. The model therefore learns to process very short and simultaneously very long sequences. This training method is very efficient since the embeddings for the tokens don't need to be recalculated and can be reused from the last training on only $n-1$ tokens of the sequence. Afterward the loss for each of these predictions is calculated and from that, the way how the model/parameters should change to decrease the loss and therefore generate the next tokens that better resemble the training data. In combination of the molecules in the string, the Transformer also takes the position of each molecule in the string into account. This is done by adding a positional embedding to the embedding of the token.

The goal of the machine learning model is that it learns the likelihood of letters in the chemical SMILES alphabet regarding all other letters that came before it in the prompt or the already generated output. This means that the model learns which preceding letters in the text are important for the determination of the next letter and which are not. In general, this is a type of Bayesian Inference where the model learns to model a probability distribution over the observed SMILES reactions, the data $\mathcal{D}$, by finding the best fitting parameters θ. This is done by computing the posterior distribution

$$P(\theta|\mathcal{D}) = \frac{p(\theta)p(\mathcal{D}|\theta)}{p(\mathcal{D})} \tag{2.1}$$

where $p(\theta)$ is the prior of the model and $p(\mathcal{D}|\theta)$ is the likelihood of the data given the models parameter. This modeling of the posterior distribution is only possible if the data

we use for training has an inductive bias which the model can learn. In other words the "only way to generalize to new input configurations is then to rely on some assumptions or preferences about the solution" (Goyal and Bengio, 2022). Some of these biases are for example "distributed representations" which shows that certain features result in patterns, "self-supervised pre-training" which states that P(X) is informative about P(Y|X). The latter can be used to argue that a model can learn the patterns in data without any classes or labels given, because the probability distribution of an input X can be informative about the distribution of another event Y given X (Goyal and Bengio, 2022). These and more general assumptions are used by the model to learn the distributions and probabilities of the SMILES language and apply them for the generation of new strings.

2.5 Transformer Architecture

The above described inputs get embedded into high dimensional vectors and combined with a positional encoding. This adds information of the position of each token in the input. The training of the transformer tunes these embeddings to map the token into a semantically meaningful embedding where vector arithmetic could be applied to. The last token of the input sequence, after the encoding, describes therefore the information of the whole input as a vector in the embedding. A vector containing an input SMILES **A** should therefore point in a similar direction as an embedding vector of a reaction **B**, as long as they are in the same reaction class or share other similar features trained on.

To generate a new token the model is "considering" the input sequence and also the already generated tokens. The architecture can be divided into two parts. The encoder and decoder. While the encoder processes the input sequence and produces embeddings for each token that contain the information of the input, the decoder uses this information and interprets it in the context of the already generated output of the model. It is important to note that the decoder only uses the embedding of the last token as any information from the previous tokens should be included in it. By doing so, the Transformer can generate an output which is not only semantically and in terms of content consistent, but also depends on the input given to the model. The model architecture is shown in Figure 2.4. The crucial principle that enables this is called attention. It occurs multiple times in the transformer architecture. Twice as self-attention to process the input sequence and the already generated output and a third time as cross-attention to combine the embeddings of the encoder with the embeddings from the self attention in the decoder (Vaswani, 2017). attention therefore acts as a way of communication between the embedding vectors for each token and enables the Transformer to focus on specific tokens in the past or the input sequence while ignoring others. So to say it can change/update the embedding vector of a token based on the information from tokens in its neighborhood. This is done many times in a row. The embedding of a token can therefore be further and further refined by its also refined embeddings of neighboring tokens, leading to a contextual embedding that contains detailed information of the whole sentence.

This is done by the self-attention, described in Equation 2.5. It starts by taking the cross product of each token embeddings by it transposed version of itself. The two products in this dot product are called the Queries and Keys. This results in a matrix with scores describing how important a token in the sequence is to another token in this sequence. Essentially, all embedding vectors that point in the same direction, so to

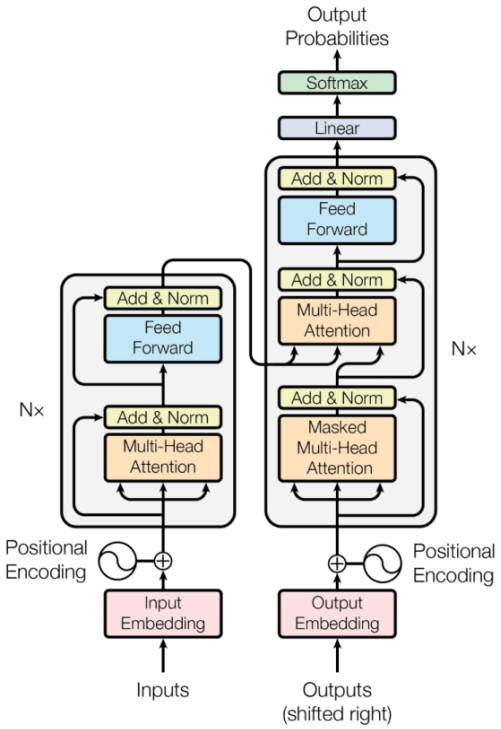


Figure 2.4: The standard Transformer architecture. [3]

speak have a roughly similar meaning result in high values in the cross product. This matrix then gets scaled through a division by the square root of the dimension of the attention block (a hyperparameter). This ensures that the softmax applied later has a steep enough gradient even though the matrix of the Query and Key could have a large magnitude (Vaswani, 2017). This property of the softmax is needed so that its variance over all tokens stays high, and no single token is disproportionately high attended to. In other words the model stays numerically stable during training.

The next step is to mask some tokens of the dot product. This can be done for causal Transformers in order to restrict each token to the information of each token that is on the left of it. Information can thereby only be used from tokens that aren't in the future of the sentence. However, this step is not included in self attention since we want to process all information from the whole sentence, while the masking is done for the decoder block, since the model should not learn to generate tokens based on future tokens that are not available.

Finally, the softmax scales all values of the Dot Product to the range between 0 and 1. This matrix therefore can be interpreted as weights of how much two tokens in the input sequence are important to each other for the generation of a new token. By a matrix multiplication with the same input sequence as the Keys and the Queries yet again, we therefore scale the values based on the attention scores described above. This describes one attention head. In order to let the model focus on different information and relations in the sequence we then conduct this attention multiple times and concatenate the results.

3. Figure from Vaswani (2017).

$$Attention(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{d_k}}) * V$$

Figure 2.5: The attention mechanism. Q, K, V, are the Query, Key and Values. Each are sequences of embedding vectors where each vector represents one token from the input sequence.

Cross attention works in a similar manner, where the Keys and the Values are the embedding vectors from the encoder and the Queries are the embeddings from the already generated output that was already processed with self attention once.

To conclude this means, that the Transformer processes the input sequence and the sequence of its own generated output via self attention, and then combines them in a cross attention block to generate a final embedding vector sequence. The last embedding vector of the sequence is then converted by a final softmax function which converts this embedding into output probabilities for each element in the vocabulary.

Sampling from these probabilities the model then selects one element of the vocabulary based on the probabilities (logits) how likely it will occur next in the sequence. This sampling can be done for example in a greedy search manner. Hyperparameters like the temperature can also influence the generation as they describe the probability of choosing a next token that doesn't have a high probability in the logit output.

Finally, the generated output is converted back from token IDs into the actual string segments of the vocabulary and combined into one string.

The Transformer architecture explained above is only the simplest structure. The model of R Irwin et al. (2022) uses either a Unified Model or a Bart model to compare the two. A Unified Transformer model uses three different pre-training types: unidirectional, bidirectional and seq-to-seq, which ensure good performance on different downstream tasks. For example the unidirectional pre-training helps to improve the performance on long text generation (Dong et al., 2019, Table 2). The unidirectional pre-training ensures that tokens can only attend to other tokens on their left in the sequence while the sequence to sequence pre-training allows the input to attend to the whole input while the output can only attend to the left context of its own sequence similar to the unidirectional version. In contrast, the bidirectional version can attend to all tokens freely. The specialty of this is that any pre-training is done on shared weights of the Transformer. This means that the model is very suitable for transfer learning, generalizes very good and is flexible in the downstream task which it used for.

2.6 Transformer vs RNNs

The advantages of a Transformer is undoubtedly the attention mechanism. This allows to select relevant information from tokens of the past in the input sequence and forget irrelevant information in the input. This can be described as an uncompressed view of the past, as the attention allows every token to communicate with every other token, slowly refining the information contained in the embedding of them. Since the multiple attention heads are not dependent on each other the Transformer can be parallelized during training. This makes the architecture efficient and scalable in regards to the training-dataset size as well as the model size.

In contrast, a RNN can't be parallelized, as the current hidden state is always dependent on the last hidden state and the current input. Therefore, the RNN must be trained sequentially since the next state can be only computed if the current one is

already known. Another big problem of RNN's is that they can't remember information from long ago in the sequence. As the next token only depends on the last hidden state and the current input, the information of the whole sequence must be contained in only the last hidden state and carried on to the next while adding the information from the current input. In contrast to the uncompressed view of the past, which the Transformer has, the RNN's context window is short and the information of past tokens gets compressed. This results in a more efficient implementation which in turn is not as powerful as the Transformer's ability to use the full input sequence.

However, for inference the Transformer needs to calculate the whole attention matrix from scratch for the generation of each next token. This makes the inference of a Transformer slow and inefficient. While the RNN can cache the last hidden state and doesn't need to calculate all previous hidden states, the transformer cant reuse old attention values. The transformer therefore scales in seq_length^2 for inference as the attention matrix has this many entries. The RNN can instead scale linearly with the sequence length.

2.7 State Space Model

This Section is mainly based on (Murphy, 2023; Gu and Dao, 2023; Gu, Goel, et al., 2021; Gu, Johnson, et al., 2021).

The state space is the set of all possible configurations of a system in a discrete space and therefore can be fully described by a number of variables. A model that can describe a sequence of configurations in a state space can for example be a simple finite-state Markov chain. In a Markov chain the current state "x_t captures all relevant information about the state of the system" (Murphy, 2023). In other words it contains all information that is needed to predict the next state in the system. This can be expressed in the following equation:

$$p(x_{t+\tau}|x_t, x_{1:t-1}) = p(x_{t+\tau}|x_t) \tag{2.2}$$

Where τ is the time ≥ 0. The probability of $x_{t+\tau}$ given all previous states of the sequence $(x_t, x_{1:t-1})$ can be described as the probability of $x_{t+\tau}$ given only the current state x_t. The probability distribution for a combined sequence in the state space for a finite length can therefore be expressed as:

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3)... = p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1}) \tag{2.3}$$

The conditional distribution of $p(x_t|x_{t-1})$ can also be described as a transition matrix $\mathcal{A}$ where each element is described by $A_{ij} = p(x_t = j|x_{t-1} = i)$ and the prior probability, which is the probability of starting from one of the available states. The matrix defines a probability for each possible transition from one state to another. In its simplest form, this system could be used to build a bi-gram language model. A very crude language model where the probability of the next character is only dependent on the last character in the sequence. The parameter, in this case the values of the matrix $\mathcal{A}$ can be estimated by the maximum likelihood estimation.

A state space model is a kind of **partially observed Markov model**. This means that the model has a hidden state z_t, that evolves over time based on the Markov

update process described above. This model can be described by two functions. The function f is used to update the hidden state based on the input and the previous hidden state and the function h to emit an observation y_t based on the current hidden state, the current input and all the so far generated outputs of the sequence. This is all dependent on the timestep t.

$$z_t = f(z_{t-1}, u_t, q_t)$$
$$y_t = h(z_t, u_t, y_{1:t-1}, r_t)$$

Where $z_t \in \mathbb{R}^{N_z}$ are the hidden states, $u_t \in \mathbb{R}^{N_u}$ are the observed inputs, $y_t \in \mathbb{R}^{N_y}$ are the outputs. q_t and r_t are the process and observation noise. As with the simple Markov model we can also describe this as a probabilistic model:

$$p(z_t|z_{t-1}, u_t) = p(z_t|f(z_{t-1}, u_t))$$
$$p(y_t|z_t, u_t, y_{1:t-1}) = p(y_t|h(z_t, u_t, y_{1:t-1}))$$

Similar to formular 2.3 which describes the unrolled Markov chain over time, we can do the same for the state space model (Murphy, 2023).

$$p(y_{1:T}, z_{1:T}|u_{1:T}) = \left[p(z_1|u_1)\prod_{t=2}^{T} p(z_t|z_{t-1}, u_t)\right]\left[\prod_{t=1}^{T} p(y_t|z_t, u_t, y_{1:t-1})\right] \tag{2.4}$$

2.8 Selective State Space Model

Another definition of the SSSM is the continuous representation. Instead of being probabilistic, this describes the SSM as a function-to-function mapping. This takes continuous signals as input and maps them into a latent hidden state from which a continuous output signal can be derived. This representation of the SSM can be defined through similar equations as the probabilistic representation, called state equation (Gu, Goel, et al., 2021).

$$z_t = \mathcal{A}z_{t-1} + \mathcal{B}u_t$$
$$y_t = \mathcal{C}z_t + \mathcal{D}u_t$$

The matrices $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ can be referred as the parameter of the model that can get trained. $\mathcal{A}$ describes how the current state evolves over time. $\mathcal{B}$ describes how the input affects the hidden state at time t. $\mathcal{C}$ describes how the current state effects the output. $\mathcal{D}$ describes how the input directly affects the output.

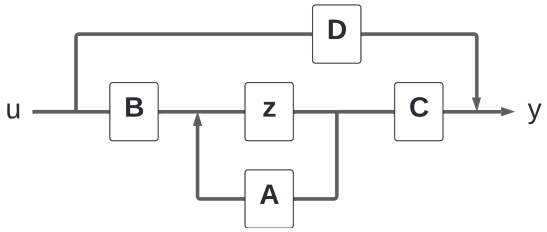


Figure 2.6: The State Space Model in continuous representation.

The above described differential equations are visualized in Figure 2.6 as a flow chart. This clearly shows for example that the matrix $\mathcal{A}$ multiplied with the hidden state z can

be used to define the next step while $\mathcal{B}$ is also used to define the next step from the input u. We can also see that $\mathcal{D}$ is used to directly describe the influence of the input to the output, ignoring the hidden state. Because of this direct connection $\mathcal{D}$ is sometimes referred as a skip connection and therefore not included in some diagrams and description of Mamba.

Since the SSM that is used as a language model gets text sequences as its input, the continuous representation is not ideal. For this reason, matrices can be discretized using the bilinear method (Tustin, 1947). This is done for the Structured Space State Model (s4) (Gu, Goel, et al., 2021) while Mamba (Gu and Dao, 2023) uses zero-order-hold. This means that every signal value gets hold until a new value is reached. This converts the discrete signal into a continuous one. The output is again a continuous signal which can be sampled in Δ long timesteps into a sequence. The discretized matrix is dependent on the continuous matrix and additionally a step-size Δ. The step-size represents the resolution of the input meaning the time a single embedding value is the value of the input signal if it were to be a continuous one. The input is therefore not a function $u(t)$ but a sequence $(u_0, u_1, u_2, ...)$, and the discretized matrices, referred as $\overline{\mathcal{A}}, \overline{\mathcal{B}}$, can be seen as transition matrices for the hidden state similar to the transition matrix for the simple Markov chain. In addition to the content of the matrices we now also learn the step-size during training.

$$z_k = \overline{\mathcal{A}} z_{k-1} + \overline{\mathcal{B}} u_k$$
$$y_k = \overline{\mathcal{C}} x_k$$

This representation as transition matrices allows for a recurrence of the hidden state z_k. This recurrent representation is shown in Figure 2.7 and has the same properties of an RNN, namely fast inference times but slow sequential training.

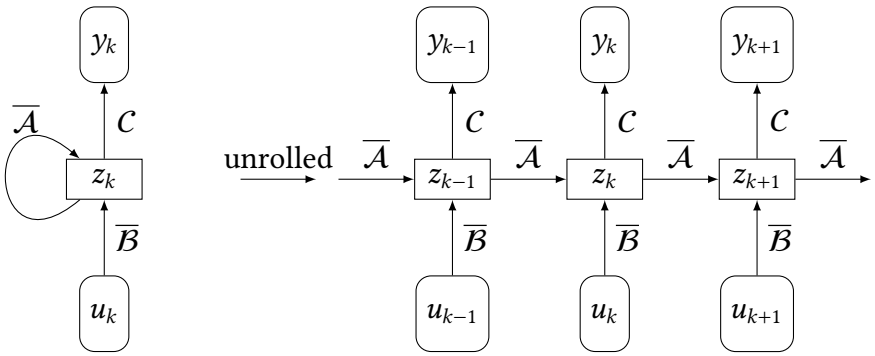


Figure 2.7: Left: Discrete-time SSM in recurrent representation. Right: Unfolded discrete-time SSM in recurrent representation.

The hidden state for each step can be calculated in the following way: For step 0:

$$z_0 = \overline{\mathcal{B}} u_0$$
$$y_0 = \mathcal{C} z_0$$

For step 1:

$$z_1 = \overline{\mathcal{A}} z_0 + \overline{\mathcal{B}} u_1$$
$$y_1 = \mathcal{C} z_1$$

For step 2:

$$z_2 = \overline{\mathcal{A}} z_1 + \overline{\mathcal{B}} u_2$$
$$y_2 = \mathcal{C} z_2$$

As the linear recurrence representation of the SSM is not parallelizable for training we can also represent the model as a continuous convolution. This enables efficient parallelization like a convolutional neural network, but also suffers from the same long inference time. This is because the convolutional filter is of fixed size and not unbounded like the recurrent structure. In the unrolling done above, each step is still defined by the last hidden state. However, one can also unroll this equation shown in Figure 2.8 to be only dependent on the input sequence u. For step 0:

$$z_0 = \overline{\mathcal{B}} u_0$$
$$y_0 = \overline{\mathcal{C}\mathcal{B}} u_0$$

For step 1:

$$z_1 = \overline{\mathcal{A}\mathcal{B}} u_0 + \overline{\mathcal{B}} u_1$$
$$y_1 = \overline{\mathcal{C}\mathcal{A}\mathcal{B}} u_0 + \overline{\mathcal{C}\mathcal{B}} u_1$$

For step 2:

$$z_2 = \overline{\mathcal{A}}^2 \overline{\mathcal{B}} u_0 + \overline{\mathcal{A}\mathcal{B}} u_1 + \overline{\mathcal{B}} u_2$$
$$y_2 = \overline{\mathcal{C}\mathcal{A}}^2 \overline{\mathcal{B}} u_0 + \overline{\mathcal{C}\mathcal{A}\mathcal{B}} u_1 + \overline{\mathcal{C}\mathcal{B}} u_2$$

This can be written as a convolution kernel which can be multiplied with an input sequence u and results in a single hidden state z that contains the information from the input sequence elements, depending on the kernel size. The kernel is defined as:

$$\overline{\mathcal{K}} = (\mathcal{C}\overline{\mathcal{B}}, \mathcal{C}\overline{\mathcal{A}\mathcal{B}}, ..., \mathcal{C}\overline{\mathcal{A}}^k \overline{\mathcal{B}}, ...)$$
$$y = u * \overline{\mathcal{K}}$$

Analogous to the 2-dimensional convolution filter used in computer vision, the SSM-convolutional-kernel can also be visualized as seen in Figure 2.8.

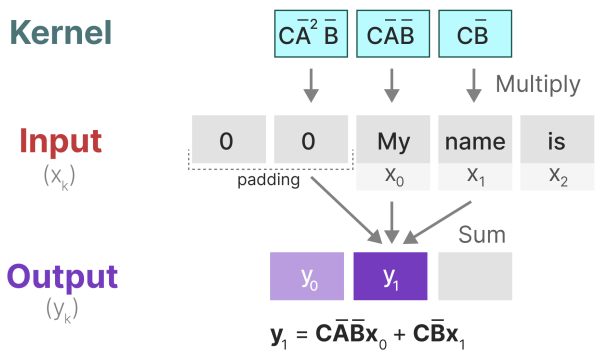


Figure 2.8: An SSM-convolutional-kernel of size 3. [4]

The big advantage of SSM and the different representation described above is, that they are interchangeable even after training. This means that one can make use of the efficient and parallelizable training of the convolutional representation during training and change to the recurrent view for inference. This can be done because all different representations have the common property of **Linear Time Invariance**. This means that the parameters of the model are not dependent on the time step and are therefore fixed regardless of the token in the input sequence. The matrices $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ are static and not dependent on the input (Gu and Dao, 2023).

From the convolutional Kernel $\overline{\mathcal{K}}$ as well as the unrolled recurrent diagram shown in Figure 2.7, we can see that the main bottleneck of the model is the matrix $\overline{\mathcal{A}}$. With increasing sequence length, we have to do repeated multiplication of $\overline{\mathcal{A}}$ (Gu, Johnson, et al., 2021; Gu, Goel, et al., 2021). A big problem of this repeated multiplication is that the hidden state, that results from the multiplication with $\overline{\mathcal{A}}$, suffers from vanishing gradients and tends to have problems incorporating long dependencies in sequences. In other words, information that was processed is forgotten over time and cant be recalled to generate the next token since the model has a limited memory horizon.(Gu et al., 2020) The solution is to use the HiPPO framework from Gu et al. (2020) to build the matrix $\overline{\mathcal{A}}$ instead of initializing it randomly.

HiPPO, short for High-order Polynomial Projection Operator, produces operators that project arbitrary functions into the space of orthogonal polynomials with respect to a measure (Gu et al., 2020). This means that HiPPO matrices can approximate the cumulative history of a continuous function on to basis functions with respect to a probability measure. Thus it is much like a Fourier transformation that can approximate a signal by decomposing it into sinusoidal functions. The difference is that HiPPO uses a sequence of Legendre polynomials as the basis function and the measure is exponentially decaying. The recently seen signal is therefore approximated very well, but the approximation gets worse the longer the time difference between the signal and the current time step is. By using this HiPPO Matrix defined in Formular 2.5 as our matrix $\overline{\mathcal{A}}$ we can capture all information of the input function u seen so far in the hidden state. Recent information is captured very well, while older information gets approximated the longer ago it was in the sequence. (Gu et al., 2020; Gu, Goel, et al., 2021; Gu, Johnson, et al., 2021) This approximation is a kind of compression of the history which gets projected into a subspace of the bounded dimension. This being the coefficients of the Legendre polynomials (Gu et al., 2020).

The HiPPO matrix used for SSM's can be defined as follows:

$$A_{nk} = -\begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & \text{if n>k (all entries below the diagonal)} \\ n+1 & \text{if n=k (the diagonal)} \\ 0 & \text{if n<k (all entries above the diagonal)} \end{cases} \tag{2.5}$$

As described above one can represent an SSM in convolutional or recurrent form and add the capabilities of modeling long range dependencies by the use of a HiPPO matrix. This special variation of an SSM is called a Structured State Space Model or S4 first introduced in (Gu, Goel, et al., 2021).

4. Figure taken from https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state, last accessed: 18.09.2024

2.9 Mamba

As said before the matrices of the S4 are not dependent on the input/time. This makes the model Linear Time invariant and enables fast and parallel computation. The downside is that the model can't perform content aware reasoning, because its matrices $\overline{\mathcal{AB}}$ are constant and not depending on the input. Tasks like selectively copying tokens from a sequence or selecting previous tokens from the history and "forgetting" others is not possible. Gu and Dao (2023) solve these issues by introducing the Mamba architecture that incorporates a selective scan algorithm into the model. This helps to selectively compress only the information from the sequence that is relevant. The Matrices $\overline{\mathcal{AB}}$ and the Δ step-size are therefore defined as a function of the input sequence. Since the efficient parallelization of the convolution is only working for constant matrices $\overline{\mathcal{AB}}$ which result in a fixed kernel, Gu and Dao (2023) implement a parallel scan operation. This computes the now recurrent convolution in parts and combines them afterwards. This brings the efficient parallelisation during training back and Mamba can now be dependent on the input and efficient to train.

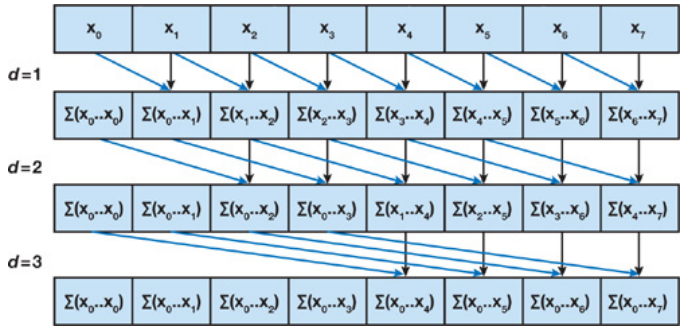


Figure 2.9: A parallel Sum scan algorithm. The Mamba architecture uses the Kernel operation shown in Figure 2.8 instead of the simple sum. [5].

Another contribution Gu and Dao (2023) brings to the previous S4 model is the hardware aware algorithm. This means that instead of copying every matrix that needs to be multiplied from the slow high-bandwidth memory (HBM) to the fast SRAM, computing the result and storing the result in HBM again, the architecture aims to skip some of the repetitive writing from and to HBM. This is done by keeping the hidden state in SRAM and only loading the SSM parameters $\overline{\mathcal{ABC}}\Delta$ directly from HBM onto SRAM. All recurrence and discretization computations are then calculated on SRAM and only the final output of the parameters is written back to HBM. This drastic reduction in memory IO's results in much faster computation similar to flash attention (Dao et al., 2022).

By storing the hidden state only on SRAM without saving any intermediate steps, the architecture cant use the intermediate hidden states for the backpropagation during training. This means that the hidden state for each step needs to be recomputed, which is still faster than storing all calculations in HBM and therefore increasing the IO operations.

5. Figure taken from Chapter 39. from Harris et al.

2.9.1 Compression Analogy

The compression of history in the input sequence introduced by the HiPPO matrix can be further visualized by the following analogy. A transformer architecture, for example, does not compress it's history because all the information from the entire sequence is used to generate the next token. The encoder uses the sequence to generate the key and values, while the decoder uses the entire input sequence plus the already generated tokens as input. This means that all the information in the sequence is used and the attention selects the important information from the sequence. This is therefore not a compression, resulting in a model with high effectiveness but low efficiency. Efficiency is measured in terms of computational cost and memory consumption during inference.

Mamba, on the other hand, is more efficient than Transformer, but sacrifices effectiveness in return. Compression is therefore a trade-off between efficiency and effectiveness, with Mamba striking a good balance between being efficient and being able to select important information from its compressed sequence input history.

The design of Mamba described in this chapter finally leads to a Structured State Space Model which is

- fast in inference through the recurrent representation (with linear complexity during inference regarding sequence length),
- parallelizable during training through the convolutional representation,
- selective in its compression of the sequence history due to the dependence of the parameters $\overline{\mathcal{A}\mathcal{B}\mathcal{C}}\Delta$ of the input,
- efficient to compute due to hardware aware algorithm,
- able to capture long-range dependencies through the use of HiPPO.

The Mamba architecture can be therefore seen as a "standalone sequence transformation" (Gu and Dao, 2023) that can be incorporated into any Neural Network, just like attention.

3
Related work

The task of retro or forward synthetic reaction prediction is often broken down into the task of predicting a single reaction of one or two reactants to a product. The second task is to use the single step prediction to iteratively build a reaction synthesis route in a retro or forward synthetic fashion. To search for a multistep reaction route, a common solution is to use a Monte Carlo search tree (Segler et al., 2018).

However, the single-step reaction task has received much more attention in recent years, as it is more important for the quality of the accuracy of the predicted reaction than the algorithm that combines the single step reactions to a full reaction synthesis tree. However this focus on the single step reaction is also being criticised by Hassen et al. (2022) and Torren-Peraire et al. (2024) who argue that the performance of a model evaluated on a single step does not fully repesent its performance in the multi step application. Since most reactions are already formulated in SMILES format and follow the rules of context-free grammars, the idea of using techniques from natural language processing is not far-fetched. Schwaller et al. (2018) and B Liu et al. (2017) use a sequence to sequence approach to model a SMILES reaction as a translation task from reactant molecules to product molecules. This is done by implementing two LSTMs. One acts as an encoder, processing the input sequence step by step and computing hidden states based on the previous token in the SMILES string and a one-hot coding of the current input token. The other LSTM acts as a decoder, predicting the probability of a product in the available vocabulary, so by sampling the output of the decoder, one can generate a product SMILES that is the result of the sequence-to-sequence translation. Schwaller et al. (2018) also use attention in the decoder. This makes the generated probability depend not only on the last computed hidden state, but also on all previous ones.

While template-based approaches use reaction templates, which "specify the atoms and bonds at and around the reaction centre before and after the reaction" (MH Lin et al., 2022), template-free methods learn the chemical rules of the transformation. This approach is therefore uninterpretable, as most machine learning models can only be treated as a black box. An intermediate approach is the semi-template based model shown in (Somnath et al., 2021) or (Yan et al., 2020). Semi-template based approaches aim to combine the advantages of both template free and template based models by splitting the templates into simpler semi-templates. Since many semi-templates are

redundant, the most common datasets such as UPSTO50K can be represented by only 170 different semi-templates (Somnath et al., 2021). These semi-templates are then used to complete them into valid reactants that result in the desired products.

However, the template-free approach uses the reaction templates only for training to learn the rules of a chemical reaction. Therefore, models introduced in (Karpov et al., 2019) or (Schwaller et al., 2019) use the vanilla Transformer architecture to translate the SMILES reactant representation into the SMILES product representation. These models use the global attention in the Transformer architecture combined with a positional encoding of the SMILES string to solve the problem of attention over large distances in the input. This is a problem that many graph-based models have. Another way to improve overall performance and generalisability is to train the Transformer on multiple objectives and multiple tasks in the chemical domain (R Irwin et al., 2022). Some of these tasks include both retrosynthesis and forward synthesis prediction, molecule optimisation, and also discriminative tasks such as property prediction and biological activity prediction. A pre-training step is also performed which utilizes Masked Molecules reconstruction to learn the statistical properties of the SMILES language. For each of the different tasks, the state of the art results of this single Transformer model show, that a good optimisation and generalisation can be achieved in the chemical problem domain.

Another way to further improve the accuracy of template-free models is to use data augmentation. This is mainly used for SMILES. The fact that SMILES are not unique for a molecule is very useful for augmentation tasks. To augment a molecule for training, it is replaced by another known SMILES representation (Tetko et al., 2020).

A problem that remains with any template-free approach is the generation of chemically invalid SMILES, or worse, the generation of syntactically invalid SMILES. Although state of the art models rank high on SMILES validity metrics with values of 5% or 9% of invalid molecules for forward or retrosynthetic valid product generation (Hu et al., 2023), there are still problems that remain. To address these, the Self Corrected Retrosynthetic Predictor (SCORP) introduced by Zheng et al. (2019) uses two different language models. One is Transformer-based and is used for retrosynthetic prediction, while the other is used to correct the syntax of the molecules generated by the Transformer. The second model is therefore based on a grammatical error prediction tool (Zheng et al., 2019). To address the same issues and additionally the problem of generating different molecules, the model of Kim et al. (2021) uses two different Transformers and cycle consistency checks to predict the reactants for a given product. This is done by training one transformer on the retrosynthesis prediction and one on the forward synthesis prediction. The process of predicting valid reactant candidates starts with the retrosynthetic prediction. The possible generated molecules are then fed into the forward predictor. This is repeated until the molecule predicted by the forward predictor is the same as the molecule used to generate possible reactants. By using this cycle, the authors ensure that both Transformers agree on the reactants and product involved in the reaction.

Another problem with many template-free and template-based methods is the lack of additional chemical information. Some approaches use energy based models and add additional information to the nodes of the molecule graphs (Sun et al., 2021, 2020). Another way of improving the accuracy by extending the training data is to transform the SMILES syntax into a grammar tree. This extends the SMILES syntax to include information such as local chemical structures and functional groups (Zhang et al., 2024).

Other models, such as the one from Zhong et al. (2022), build root-aligned SMILES from normal SMILES. This allows a one-to-one mapping between the reactant and product representations. This reduces the distance between the tokens that need to be changed by the reaction and therefore helps the model to focus on the chemical reaction rather than the syntax of the SMILES language.

As most of the models presented so far use SMILES as their representation for the molecules, they are all based on more or less the same structure and may therefore have similar problems or advantages. However, some approaches use a graph representation because molecules are the optimal data to represent in a graph. This can be done by representing the nodes of the graph as atoms of the molecules and the atomic bonds as edges in the graph. To distinguish between different types of atoms or bonds, a graph can use several classes of edges or nodes. For example, to represent carbon or oxygen atoms, or single or double bond types (Z Lin et al., 2023). Some of these models also use both SMILES and graph representation to combine the advantages of both methods (Seo et al., 2021). This is done by injecting local information about the graph structure into the SMILES representation within the a blocks, while still using a SEQ2SEQ approach.

The conversion of graph structures into SMILES strings is another approach with promising results. This aims to incorporate the permutation invariant graph representation into attention based sequence prediction. This helps to combine the information in the local reactive region, using a graph representation, with the information in the global reaction context, appropriated by the attention mechanism (Wan et al., 2022; Tu and Coley, 2022). In order to be able to introduce data augmentation into the favourable graph representation, Hu et al. (2023) annotate the graph structure with indices to be able to augment it. This is useful because it ensures that the graph is compatible with the index-based representation of SMILES.

Adding data to the model can not only be used to introduce information in two dimensional space. Yao et al. (2024) use 3-dimensional conformation data about the molecules and adds it to the graph representation. This is then used in a Transformer that can take graph structures as input and is supported by atomic mapping between reactant and product. This leads to the ability to order the graph by its nodes and therefore also to generate the product node by node in an autoregressive manner. This node aligned generation is described in Figure 3.1.

The encoder of the NAG2G model uses 3D conformation, 2D graph and chemical information to embed the product of the reaction. This SOTA embedding is then used by the decoder together with the node aligned generation process to generate the reactants of the reaction. As illustrated in Figure 3.1, node alignment ensures that all atoms of the molecule that are not added or changed by the reaction are at the beginning of the sequence. The newly added atoms are then appended to the end. This requires the model to repeat most of the sequence in exactly the same order, generating only the new atoms that are added. This is much easier than generating a completely new SMILES that is different from the input SMILES, even though only a few atoms have changed.

The design of the NAG2G model is promising if the Mamba architecture can be used effectively. For one thing, the atom mapping ensures a meaningful order of atoms that is the same for encoding and decoding. This fits very well with the linear generation process of Mamba, which does not use positional encoding. Secondly, node alignment ensures that the reaction centre (the part where the reaction takes place and atoms are changed or added to the molecule) is always at the end of the sequence. See Figure 3.1 for a visualisation of the decoding. This is very suitable for processing with the Mamba

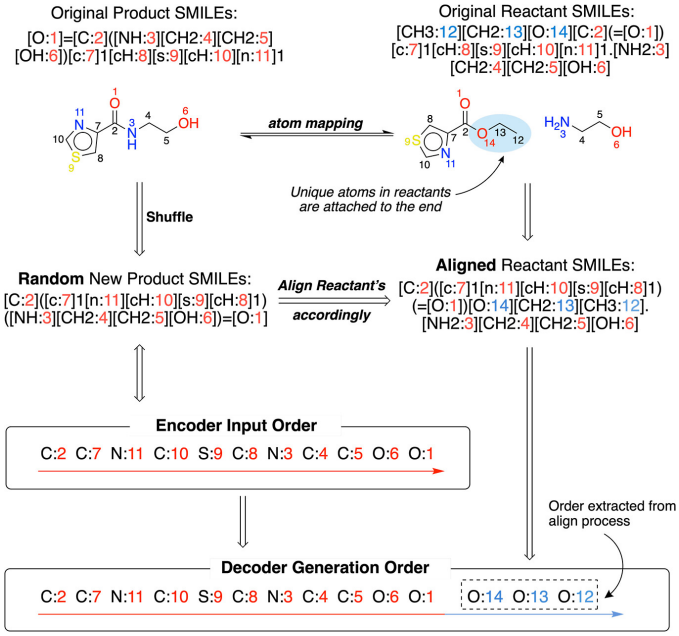


Figure 3.1: NAG2G model generation process using node aligned atom mapping [1].

architecture, since Mamba performs a compression of the sequence history. As seen in 2.9.1. This means that the information processed at the beginning of the input is not as present as the information at the end of the sequence. The reaction centre, which is arguably the most important part of the molecules, is therefore at the end of the sequence, where Mamba can best select and recall its information. This design could ensure a good selectivity of the Mamba model for the important information in the reaction.

1. Figure taken from Yao et al. (2024).

4 Methods

As R Irwin et al. (2022) have shown in their work, pre-training can significantly improve the performance of a large language model for chemical tasks. The Chemformer Model has been pre-trained with molecular mask prediction and then fine-tuned for many other objectives such as reaction prediction, molecular optimisation or property prediction. The result is a language model that is generalised to the chemical domain, understands the SMILES language and can perform multiple chemical tasks. However, a disadvantage of using a Transformer is the long inference time required to generate molecules. The selective state space model Mamba is known for its fast training and inference times, while maintaining the long context window and selectivity to forget irrelevant information in the past. In addition, Gu and Dao (2023) have shown in their work that Mamba is able to perform very well on a DNA classification task, which requires a long context window and selective selection from the query history. This could imply a similarly good performance on chemical tasks which are comparable to many biological tasks. Therefore, this thesis attempts to exploit the advantages of this model for the same tasks to which the Chemformer has been applied. This work aims to compare the two architectures in terms of reaction prediction task performance and inference time measurements, as well as their scalability in terms of batch size and model size. Furthermore, an ensemble model is developed, which aims to combine the two pre-trained models and leverage its generalization which is shown to be slightly disjunctive in the prediction of SMILES. This means that both models learn to correctly predict different parts of the dataset, while incorrectly predicting others that are correctly predicted by the other model. This suggests a good performance of an ensemble model, which is further supported by the results of Dao and Gu (2024), which found that the combination of attention and Mamba blocks into one model improved overall performance.

4.1 The Benchmark Datasets

The dataset used for pre-training is the Zinc dataset (JJ Irwin et al., 2012). It contains over 750 million purchasable chemical compounds in SMILES format and additional chemical information such as 3D conformations.

The dataset used for fine-tuning is the USPTO-50k dataset. It is a subset of processed chemical reactions from US patents from 1976 to 2016. It contains 50k of randomly selected reactions that are classified into 10 reaction classes. All reactions are given in SMILES format and reactants, reagents and products are separated by a ">" character. It contains 10 different reaction classes, which are not evenly distributed. See Figure A.1. An example of a file containing the dataset can be found in Figure A.2.

4.2 The Benchmark Model

There are many models of different architectures that can achieve near SOTA results in the task of retrosynthetic reaction prediction. Recent papers (Yao et al., 2024; Z Lin et al., 2023; Somnath et al., 2021; Hu et al., 2023) show that graph-based structures tend to capture the necessary information for reaction prediction the most efficiently, but are difficult to scale and slow.

To allow for a good comparison with the Mamba architecture on these tasks, the thesis chooses a pure transformer architecture that predicts reaction reactants in an autoregressive manner (R Irwin et al., 2022), called Chemformer. The Chemformer model is a good comparison to other language models because it solves the task using only the SMILES input and no additional information. This ensures that the architecture has the same information to train as the Mamba model, which also trains only on the SMILES sequence inputs. In addition, the Chemformer is trained on many chemical tasks, making the model versatile and a good comparison for many where it achieves SOTA or near SOTA results. However, an approach to add additional chemical information could improve performance. More information on such an approach can be found in Chapter 3 and 4.4.

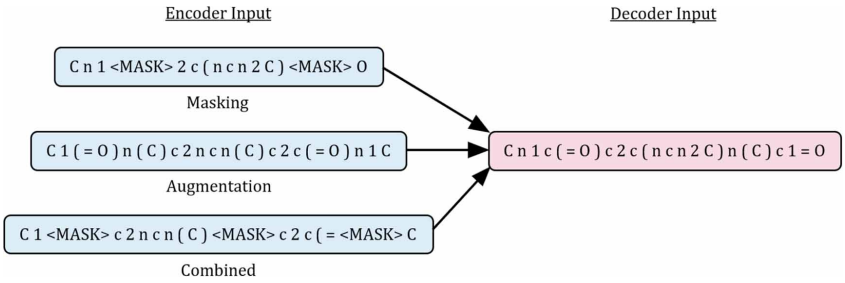


Figure 4.1: Diagram explaining the masked pre-training of the Chemformer Model. [1]

The Chemformer is pre-trained on 100M molecules from the ZINC dataset (JJ Irwin et al., 2012) using masked language modelling, which is visualized in Figure 4.1. This provides a broad understanding of chemical space, which is important for good performance in many downstream tasks. Fine-tuning is then performed on a variety of different chemistry tasks such as reaction or property prediction or molecule optimisation. See Figure 4.2 for a explanation of the fine-tuning task.

The USPTO-50k dataset has been divided into training, validation and test sets of 40K, 5K and 5K samples respectively.

1. Figure from R Irwin et al. (2022).

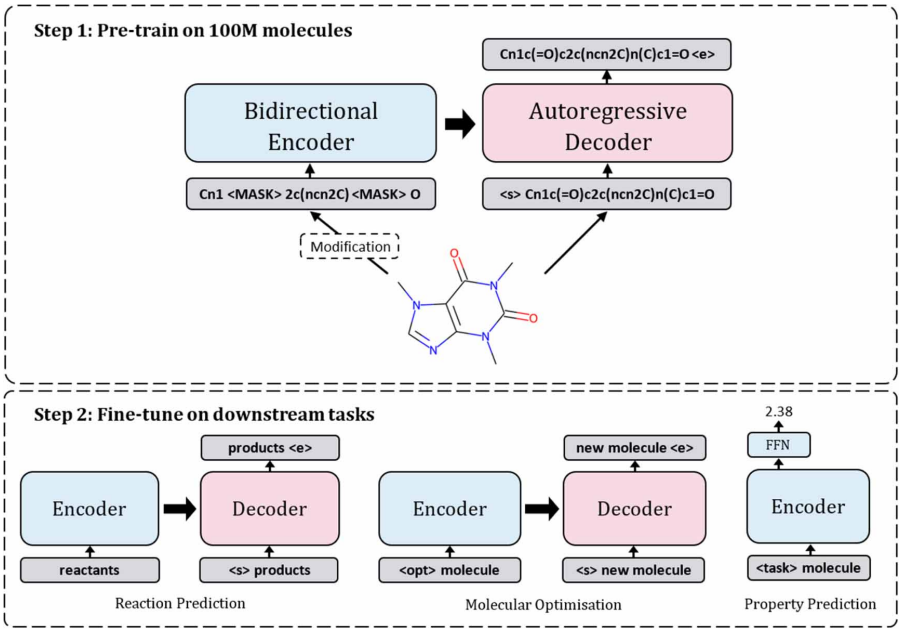


Figure 4.2: Diagram explaining the different fine-tuning tasks used for the Chemformer. [2]

4.3 Model Comparison

To ensure a fair comparison of the two models, the Mamba implementation was made as similar as possible to the Chemformer implementation. This meant using the same tokenizer and vocabulary, exactly the same dataset and its training, test and validation splits, as well as other hyperparameters common to both models. For example, the maximum sequence length, the decoding algorithm and the number of epochs to be trained. To compare the performance and inference time in terms of model size, the batch size, number of heads and number of layers of both models were chosen so that the number of parameters of both models was approximately the same. In this way, plotting the results against the number of model parameters results in evenly distributed data points and therefore a better visual comparison, which can be seen in Figure 5.1. The comparison of the two models was performed at two levels of detail. The simplest and less detailed analysis is to simply compare the performance of the two models for the given metrics of the task. In the case of this thesis, these are the molecular accuracy, the token accuracy and the invalid percentage.

4.3.1 Molecular Accuracy Metric

The molecular accuracy is the most important metric. It is defined by the proportion of correctly predicted SMILES strings in the test-dataset. This can be interpreted as the overall and generalizability of the model in the chemical domain. It is used as a benchmark metric for the reaction prediction task.

2. Figure from R Irwin et al. (2022).

4.3.2 Token Accuracy Metric

The second metric is token accuracy. This is defined as the fraction of correctly predicted tokens in a SMILES string, taking into account the position of the token. This fraction is then averaged over all SMILES in the test dataset to represent the overall token accuracy. Unlike molecular accuracy, this fraction can be high even if the final SMILES are incorrect. This is due to the sensitivity of the SMILES language to small changes. While the molecular accuracy would count such SMILES as a wrong prediction, the token accuracy can give more detailed information about how many tokens were actually correctly predicted, i.e. to what extend the model failed to predict that molecule. Because of this finer-grained analysis of the model, this metric is a good indicator to use during training, for example, since most models need to be trained for a few epochs before molecular accuracy starts to show progress. The ability to return percentages at the level of individual SMILES also makes it very good for rapid updating of the training process, as values can be recorded even between completed epochs.

4.3.3 Invalid Percentage

The third metric used to evaluate the model is the invalid rate. This is defined as the proportion of molecules in the dataset that are chemically incorrect or unstable. This metric does not include the actual comparison with the ground truth of the dataset, therefore it only analyses the ability to learn the chemical rules and to generalise on the chemical space.

In order to make use of the metrics described above, this thesis compares the metrics with the number of model parameters. Since the performance is largely dependent on the size of the model.

4.3.4 Inference Time Measurements

To assess another aspect of scalability, the time taken by both models for inference is a good indicator. A simple measure that can be used is the average time it takes to predict a SMILES in the dataset A.5. However, this does not take into account the dynamic length of the SMILES in the dataset. To remove the SMILES length factor from the measurements, token throughput is a better metric. This is the number of tokens generated by the model per second. This is compared again for different model sizes to show the influence of the number of parameters on the inference time. While the token throughput is also dependent on the batch size, the thesis also reports the token throughput against the batch size used during inference, averaged over all model sizes. This can further visualise the scalability as the model size factor is not eliminated. This evaluates the inherent scaling characteristics of the model, which is induced by the architecture, rather than the number of FLOPS required for inference (which is more dependent on model size).

To make the above measurements as robust as possible, the inference was run 5 times and all values were averaged across the different runs. In addition, a single GPU was used for all predictions and all other activity on that GPU was reduced as much as possible.

4.3.5 Chemical SMILES Evaluation

The second level of comparison of the two models involves a detailed analysis of the SMILES generated for each model. This evaluation focuses more on the chemical aspects of the generated SMILES and their linguistic features. For example, the accuracy of the SMILES string compared to the SMILES length (roughly equivalent to the size of the molecule), the comparison of the different reaction types used in the dataset and also the comparison of the generated SMILES from the different models.

These evaluations have to deal with the unbalanced nature of the test data set. While this is not a major issue for a chemically based expert system, each language model may have different performance depending on the length of the input. This is due to the fact that longer input strings require more context and longer memorisation of information. The further apart the information is in the input sequence, the harder it is for a language model to reason about that information. While the Transformer architecture is very capable of selecting the important information in a long sequence, the Mamba model, due to its linear nature, is not as capable of performing this task. That's why the length of SMILES is an important feature when comparing natural language models for chemical tasks.

Another feature of the dataset is the reaction type for each reaction in the dataset. These are abbreviated and counted in ten different reaction classes. This is because the different reaction classes can vary in their complexity and also in their complexity when represented in SMILES syntax. We can also compare the performance of the two models for each reaction class. This evaluation can reveal different generalisations of the chemical reaction space that a model may have learned, and also show which reactions are feasible for SMILES-based processing.

Another way of showing the differences between the models is to compare their predictions in a confusion matrix. This means counting the SMILES that both models predicted correctly, the SMILES that both models failed to predict and, most importantly, the SMILES that one of the models predicted correctly and the other failed to predict. This can further highlight any differences between the models and provide insight into which architecture can better generalise to the dataset. The evaluation of this overlap between the models can also be done for each reaction class or each SMILES length to show the differences in the models as a function of these two factors.

Furthermore, the evaluation of incorrectly predicted SMILES is also of great importance. Similar to the token accuracy metric, we can compare the incorrectly predicted SMILES to the ground truth by calculating the Tanimoto similarity. This is a similarity value based on chemical information that is close to 1.0 if the two molecules are chemically similar and close to 0.0 if they are not. This can give more insight into how the model failed to predict some SMILES. This is a complement to the token accuracy metric, which focuses more on the chemical aspects than on the linguistic syntax of the SMILES language, since its syntax may hide important information about model performance in relation to chemical structure.

4.4 Ensemble Model

An ensemble model of several models can be designed in many ways. To keep the ensemble model as lightweight as possible, the option of training a new model to decide which prediction from which model to use was not explored further. Instead, the focus of this chapter is to explain the different ways in which an ensemble model has been constructed and tested using only the predicted log probabilities of both models.

The simplest way to construct such an ensemble is to always take the prediction of the model that is most confident in its prediction. This involves adding together all the log probabilities of the tokens in the sequence. Due to logarithmic laws, this results in a multiplication of the probabilities in linear space. The result is the overall conditional probability for the whole sequence.

An alternative approach to utilizing the logits in an ensemble model is during the decoding step, as outlined in Figure 4.3. In this approach, the two models are executed in parallel on the same input, generating the log probabilities for each token in the alphabet at the subsequent sequence position. Subsequently, both of these logits are averaged, resulting in a new logits vector that is employed to determine the subsequent token in the sequence. This modified logits vector and the new sequence are then utilized for the subsequent step of the decoding process until the maximum sequence length is reached.

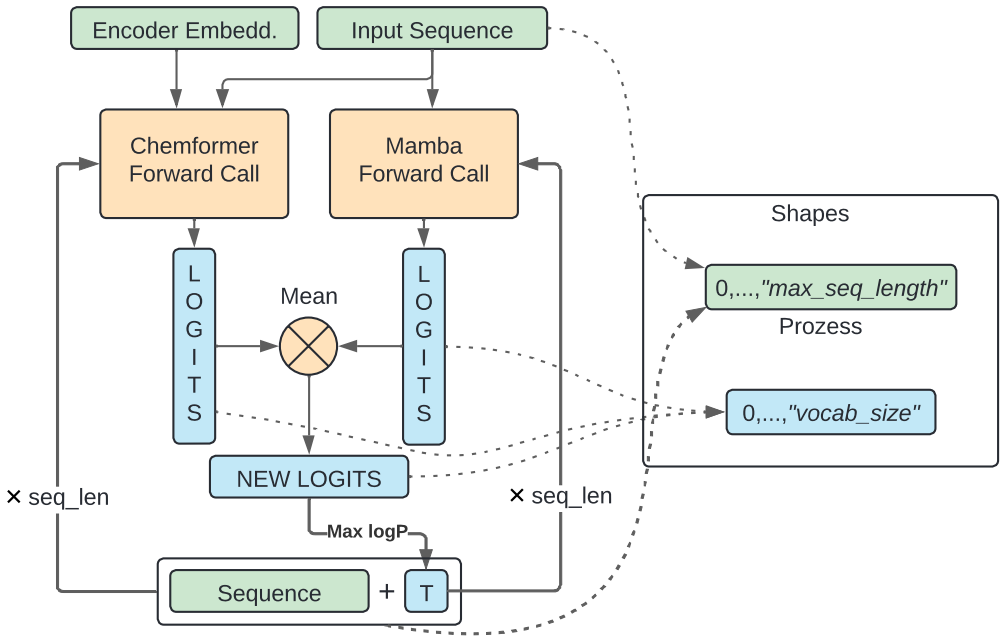


Figure 4.3: Diagram explaining the merged decoding process of the two base models. Note that the "Mamba Forward Call" uses the "Input Sequence" only for the first step of the decoding. Any further tokens are only generated based on the last one and the internal state of the model (The model's structure in can be seen in Figure 2.6).

4.5 Mamba and NAG2G Mixture Model

A general introduction, which is required for this section, can be found in Chapter 3. Incorporating the Mamba architecture into the NAG2G training presents some challenges. The incorporation of the 3D and chemical information into the encoder is closely linked to the attention mechanism. It is therefore not possible to simply replace the Transformer with a Mamba model. Figure 4.4 shows the simplest way to combine the NAG2G encoder architecture with the Mamba architecture as a decoder.

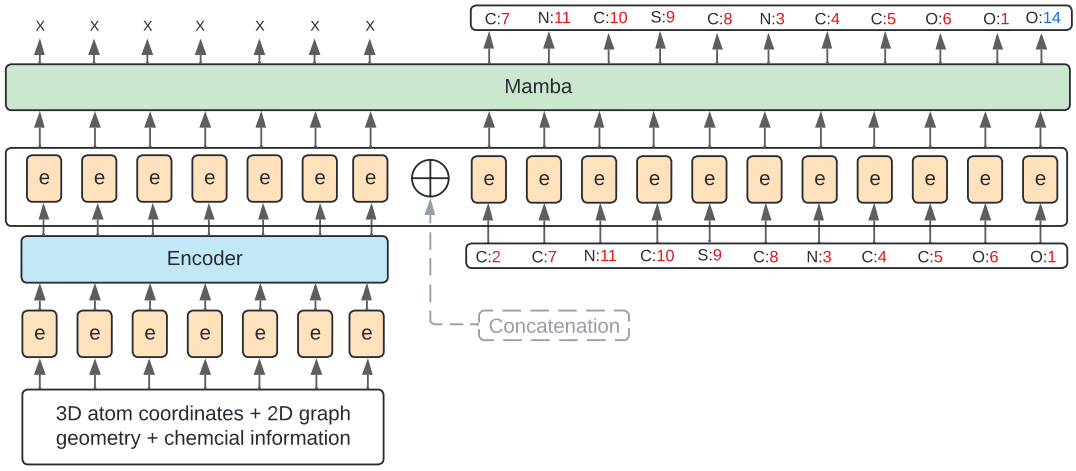


Figure 4.4: Simplest suggestion on how the NAG2G encoder can be fused with the Mamba model to increase the NAG2G efficiency while preserving SOTA performance.

5 Evaluation

All evaluations for retrosynthetic reaction prediction that are done in this chapter are without a known reaction class.

5.1 Molecular/Token Accuracy

Training both the Chemformer model and the Mamba model with different model sizes shows that Mamba can be competitive with Transformer on this task. As Figure 5.1 shows, both models scale similarly in performance, with the accuracy slowing plateauing for models with 50 million parameters. Models bigger than that show only insignificant performance improvements, even with large differences in the model size. For small model sizes Mamba actually adapts better to the task and achieves higher accuracy, but for larger sizes Transformer beats Mamba by about 4%. From Figure 5.1 it is also clear that the replication of Chemformer was successful, as the Chemformer results from the paper (blue line) seem to connect to the replicated results in green. Comparing the two Figures 5.1 and 5.2 we can see that Mamba performs better for most model sizes after the finetuning. Before the finetuning this is only true for the two smalles models. This shows that specially the smaller Mamba models, compared to the larger models, adapt very well to the task. Presumably this effect is also true for the pre-training, but occurs occurs much less and only for even smaller model sizes. One difference between the results after pre-training and after fine-tuning is that model size has a strong influence on Transformer's performance. The plateau sections in the sequence of transformer models with increasing model size are models with the same embedding size but increasing number of layers. This shows that the pre-training is sensitive to the hyper-parameters of the models, and in particular the number of layers alone cannot further improve the performance after a certain imbalance between the embedding size and the number of layers.

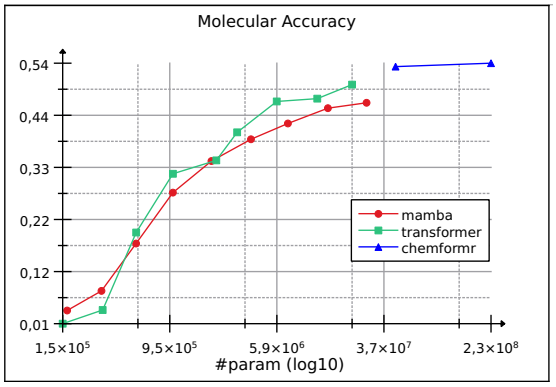


Figure 5.1: Molecular accuracy for Mamba and Chemformer models in different model sizes **after fine-tuning** to predict retrosynthetic reactions. The non-replicated results of the Chemformer-Large from the paper are added in blue.

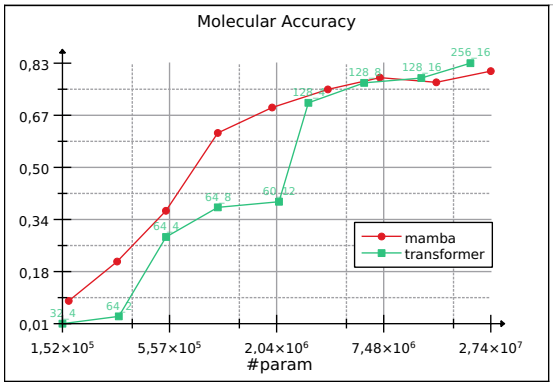


Figure 5.2: Molecular accuracy for Mamba and Chemformer models for different model sizes and **after masked pre-training and without finetuning**. At each data point for the Transformer, the model hyperparameters "Embedding Dimension" and "Number of Layers" are given, separated by an underscore. The high preformance is due to the task beeing *masked prediciton* and not *reaction prediction*.

As mentioned above, the advantages of the Transformer only seem to pay off on larger models. Therefore, Mamba performs better on smaller models. This thesis is further supported by Figure 5.3, where the same pattern can be observed.

The invalid rate also supports this thesis. Figure 5.4 shows that for larger model sizes, fewer molecules generated are chemically invalid, and Transformer again performs slightly better than Mamba, especially for larger model sizes. Both models have a very low invalid rate of less than 1%. This shows that template-free architectures are capable of modelling chemical space without handwritten rules.

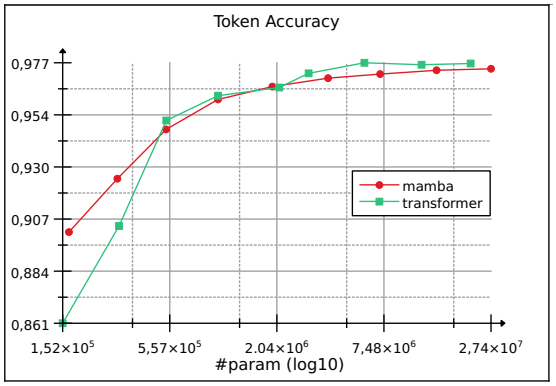


Figure 5.3: Token accuracy for Mamba and Transformer after the fine-tuning.

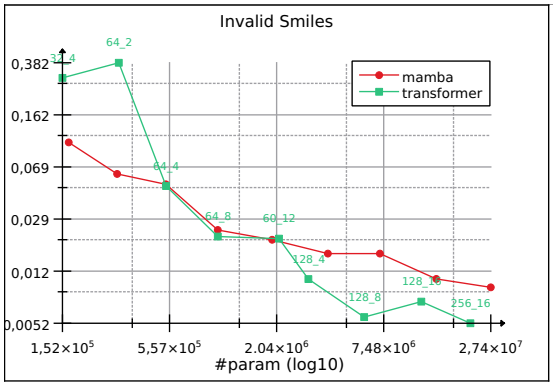


Figure 5.4: Chemically invalid rate of generated molecules for Mamba and Transformer after the fine-tuning.

Figures for performance after pre-training can also be found in Appendix A, reinforcing the findings above.

5.2 Inference Time Measurements

The inference time measurements clearly show the big advantage of the Mamba architecture, which is the speed in the inference mode. While the accuracy of the model is roughly the same and scales equally, the inference time is not. The transformer is much slower, and compared to Mamba, the inference time also increases rapidly as the model size increases. See Figure 5.5 where the token throughput (number of tokens generated per second) decreases with model size. This is because larger models take more time to generate the same number of tokens. In contrast to pre-training, where only the Transformer architecture showed a large dependence of accuracy on model hyperparameters, the inference time for both Mamba models also suffers from this. In fact, the inference time for both models is highly dependent on the number of layers. While Mamba scales better with model size, the effect of the hyperparameters is also smaller than for the Transformer. This can be seen, for example, in the huge jump in speed for the Transformer from a model with 12 layers and an embedding dimension of 60 to a model with an embedding dimension of 128 but only 4 layers. The overall insight from Figure 5.5 is that token throughput for Mamba models decreases much more slowly than for the Transformer, leading to an increasing difference in inference time with larger models. For the largest model evaluated, Mamba is about **379%** faster.

Another factor that has a huge impact on the throughput of a model is the batch size. This can be seen in Figure 5.6, where the token throughput is plotted for different batch sizes, but averaged over all model sizes. This shows that regardless of model size, both Mamba and Transformer can process and generate tokens faster with larger batch sizes. This can be explained by the parallel implementation of the attention and Mamba modules, which process the input sequence in batches. Again, the Mamba architecture shows inferior scaling characteristics.

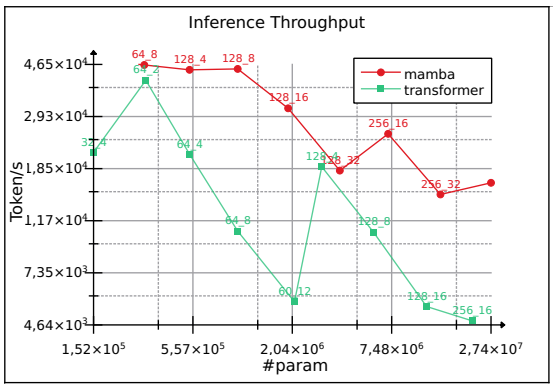


Figure 5.5: Inference throughput measured in tokens per second for different model sizes of Mamba and Transformer architectures. All measurements were taken for 8 different batch sizes and averaged to eliminate their factor.

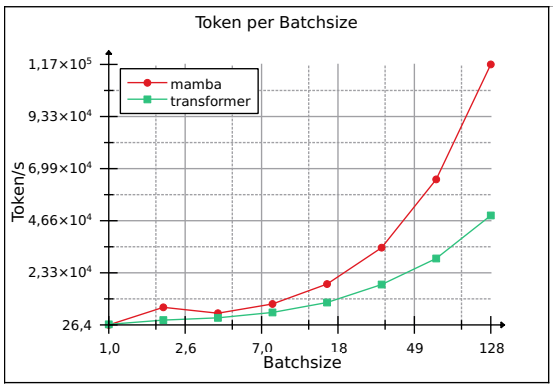


Figure 5.6: Token throughput measured in tokens per second for 8 different batch sizes. All measurements were taken for 8 different model sizes and averaged to eliminate their factor.

5.3 Chemical Analysis

When comparing the Chemformer and the Mamba architecture, an important aspect of the evaluation is the chemical and representational properties of the molecules

generated. One of these is the length of the generated SMILES and the performance of the model on SMILES of a given length.

Figure 5.7 shows a cumulative graph of the number of correctly predicted SMILES sorted by their length. The better performing Transformer therefore achieves a higher cumulative number of correctly predicted SMILES. It can also be concluded that both Mamba and Transformer perform well on medium long SMILES. This can be inferred from the fact that the highest gradient is in the bottom third of all SMILES lengths observed. The plot also shows that Mamba and the Transformer model perform quite similarly in terms of which exact SMILES are correctly predicted. This can be inferred from the very similar shapes of the gradient lines. Both have local maxima and minima at exactly the same SMILES length. However, Figure 5.7 gives an unbalanced view of the dependence of SMILES length on model performance. The blue line shows the distribution of the test dataset. We can see that most SMILES are correctly predicted where the dataset has the most SMILES of that length.

Despite this imbalance, the theory above can be verified by plotting molecular accuracy or token accuracy against SMILES length (Figure 5.11 and 5.12).

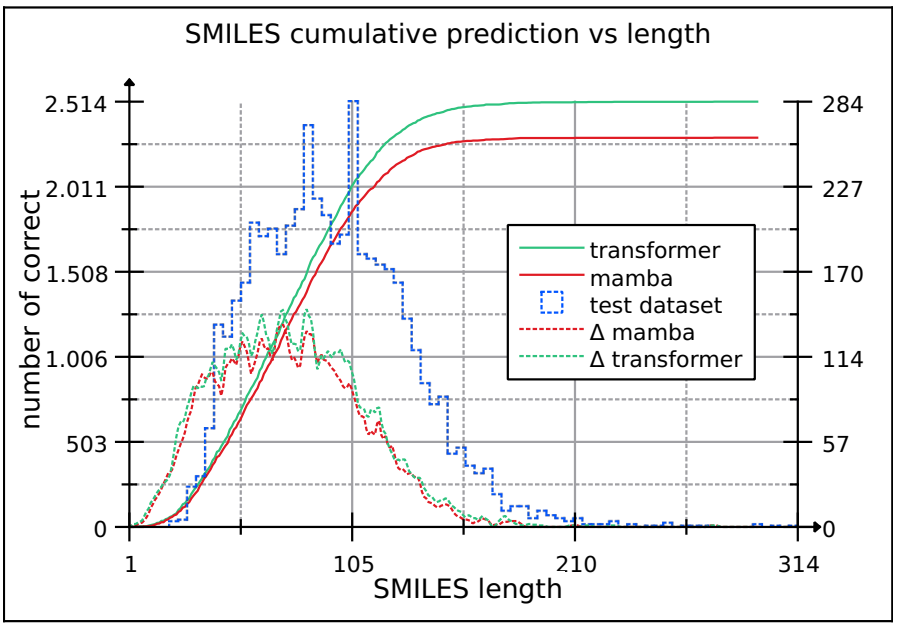


Figure 5.7: Cumulative plot of the number of correctly predicted SMILES, sorted by their length. The length is defined by the length of the reactants and the product. Additionally, the distribution over the test dataset is plotted in blue.

Figure 5.11 shows the molecular accuracy for 30 different bins of SMILES length. In contrast to 5.7, it can be clearly seen that Mamba does not perform well on SMILES longer than 200 tokens, while the Transformer also drops in performance for longer SMILES, but manages to outperform Mamba by a large factor. This could be explained in several ways. Firstly, Mamba may have problems capturing longer dependencies in longer input sequences. This is likely to be the case because any linear sequence model has to compress the history of the sequence, making it harder to capture long dependencies. This analogy is explained in more detail in Section 2.9.1. However, Dao and Gu (2024) have shown in their work that Mamba performs exceptionally well on longer sequences, as they have successfully trained Mamba on long dependency tasks from the biological

domain. An overall performance drop for longer context sizes is to be expected (NF Liu et al., 2024), which is usually found for much longer context sizes than this task.

Therefore, secondly, it is very likely that especially longer examples of SMILES syntax are difficult for language models to process. This cannot explain the discrepancy between the Mamba and Transformer scores for longer SMILES, but it can explain why both models perform worse on longer SMILES (Figure 5.12). As the SMILES syntax can become very complex for larger molecules, it is probably also harder for any model to generate correct molecules. An example are chemical cycles, which contribute a lot to long SMILES and have to be split into a linear sequence within the SMILES. This and other problems with the SMILES syntax are discussed in Section2.2 This makes the SMILES representation less than ideal for such structures. This theory is further supported by Figure A.6, which shows that the invalid rate increases drastically with longer SMILES. This however needs some consideration, because the bins for longer SMILES only contain a few SMILES making it hard interpret these values. The number of chemical rings in a molecule, which is another indicator of the complexity of a SMILES, increases as expected with SMILES length. This is shown in Figure 5.8, where we plot the average number of rings against the length. This is expected as larger molecules

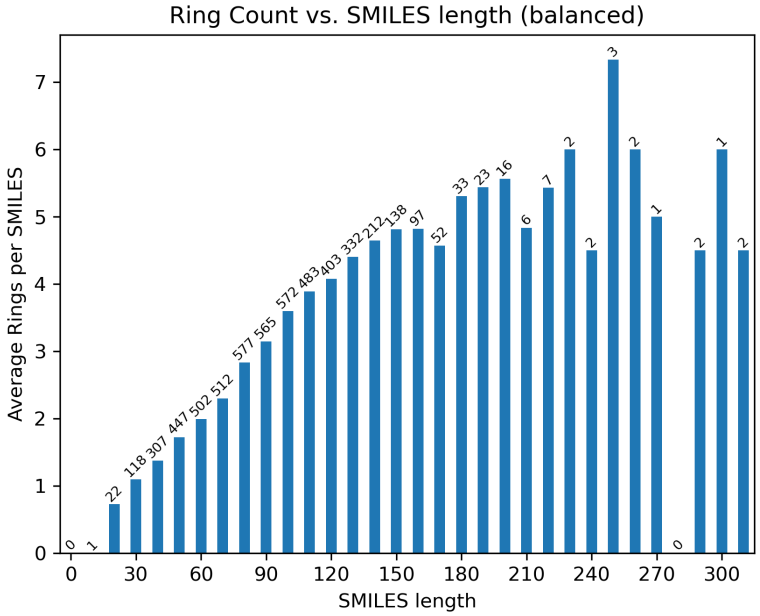


Figure 5.8: Number of chemical ring formations in the reactants of the USPTO-50 test set.

tend to have more complex structures. Conversely, rings themselves contribute many tokens to a molecule, so molecules with rings automatically become longer. We can therefore argue that an increasing number of rings for larger SMILES could indicate that the SMILES syntax is not ideal for molecules with ring structures to be processed by language models. This could be another factor why the two models perform worse on longer SMILES. This can also be seen in Figure 5.12. The token accuracy shows the percentage of correctly predicted tokens. This is independent of molecular accuracy, where the generated SMILES must be correct. Since one wrong token is enough to make the SMILES incorrect, the model could actually have done a very good job of predicting most of the molecule correctly, but fail for only one token. The comparison of Figure

5.11 and 5.12 shows that both models lose performance in token accuracy with longer SMILES, while only Mamba performs significantly worse in molecular accuracy. This suggests that the Chemoformer might produce different SMILES, resulting in lower token accuracy, but still describing the correct molecule. This is possible because SMILES are ambiguous (Section 2.2). A molecule can be described by several SMILES. This could be an explanation for the different performance shown in Figure 5.11.

Another explanation for the lower scores on longer SMILES is the imbalance of the test data set in terms of molecule length. This is shown by the bar labels in Figure 5.11 and 5.12 and A.6, which show the total number of SMILES in this bin. This imbalance is not ideal for comparing language models, since the length of the context is an important aspect in evaluating performance. However, to allow a fair comparison with the original Chemformer, the same splits had to be chosen. Because of this imbalance, the scores of the bins with fewer data points need to be treated with caution. These bins correspond to the bins with poorer performance of the two models, so their interpretation must be treated with caution.

Finally, another reason for decreasing performance on longer SMILES may be an uneven distribution of reaction classes in the test data set. This can be problematic as different models may perform differently on reaction classes. Figure 5.9 shows that not every reaction class produces all molecule lengths. In fact, only RX_1, RX_2 and RX_6 have example reactions with more than 300 tokens. Therefore, it might be possible for a model to perform well on long SMILES in this test dataset by specialising in just these three reactions, without generally performing well on all long reactions.

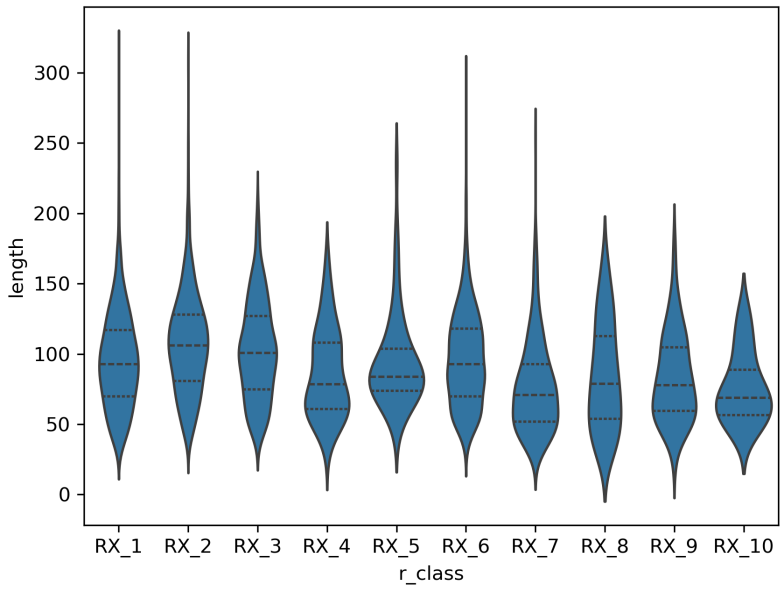


Figure 5.9: Violin plot of the reaction classes against the SMILES length in the USPTO-50 test set. SMILES length is a number of tokens of the combined product and reactants SMILES.

A set of example reactions sorted by length can be found in Figure 5.10. This shows the three longest and three shortest reactions in the dataset. The atoms for all predictions from the Mamba and Transformer models are highlighted in green and red, indicating how much that atom contributes or does not contribute to the similarity index. This can

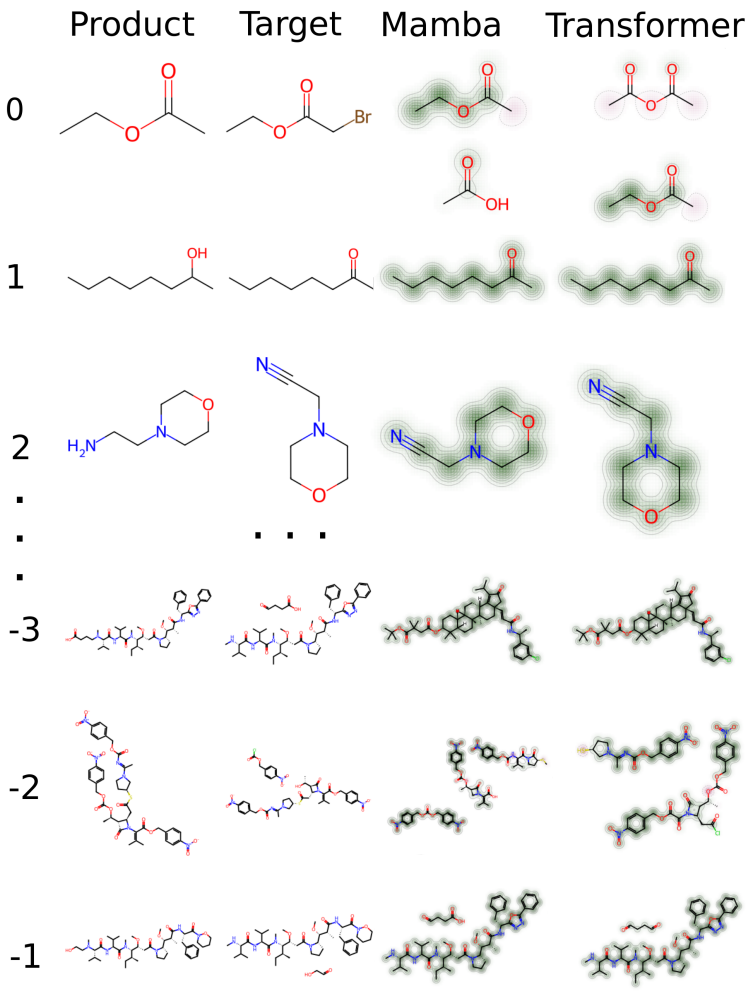


Figure 5.10: The three shortest and three longest reactions in the USPTO50 dataset. The shortest are labeld with 0,1 and 2 and the three longest with -1, -2 and -3 to be consistend wiht common list indexing methods. For the Mamba and Transformer predictions of the reactants each atom is highlighted how much it positively (green) or negatively (red) it contributes to the similarity index between the target and the respective model prediction.

be used to quickly visualise which parts of a molecule have been correctly predicted. From Figure 5.10 it is clear that the shorter reactions also contain the simpler molecules. For the second longest reaction (-2) one can clearly see that most atoms are predicted correctly, but Mamba splits the product into 3 molecules, whereas the target consists of only two. The Transformer prediction does this correctly, but a few atoms are predicted incorrectly, leading to an incorrect reaction prediction.

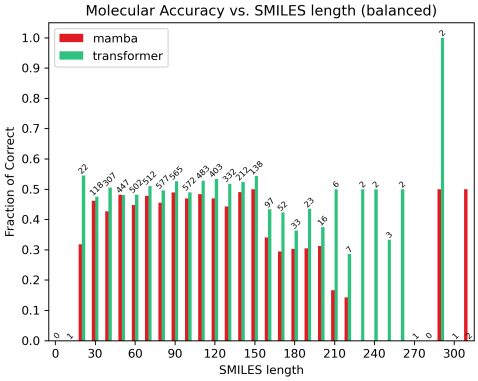


Figure 5.11: Histogram of molecular accuracy on SMILES length. The labels on each bar are the total number of reactions in that bin.

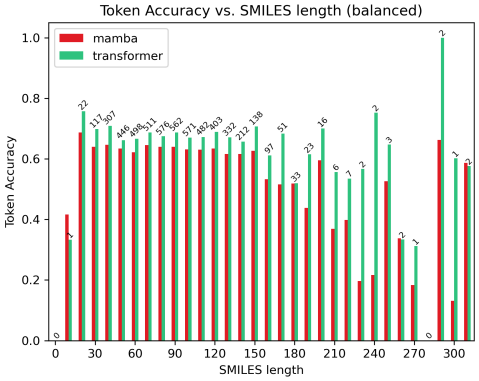


Figure 5.12: Histogram of token accuracy on SMILES length. The labels on each bar are the total number of reactions in that bin.

While both models struggle with longer SMILES, Figure 5.13 shows a different view. This figure plots the SMILES length against the Tanimoto similarity between the prediction and the ground truth. This is done for all incorrectly predicted molecules, as the Tanimoto distance for the correct predictions would be 1. The Tanimoto similarity is a measure of how similar two molecules are to each other. The linear regressions show a positive trend for each reaction class, indicating that the similarity of the prediction to the ground truth is higher when the SMILES is longer. This can only be said for those reactions that were not correctly predicted. The number of correct molecules still decreases with the length of the SMILES. The low similarity of some very short SMILES could be explained by the fact that the Tanimoto similarity is defined by the ratio of the intersection and union of two molecular fingerprints. In general, the smaller the molecule, the more influence a single wrong bit of a fingerprint has on the ratio. Smaller molecules can therefore vary much more in their Tanimoto similarity scores than larger ones. This size dependence is also described by Willett (2006).

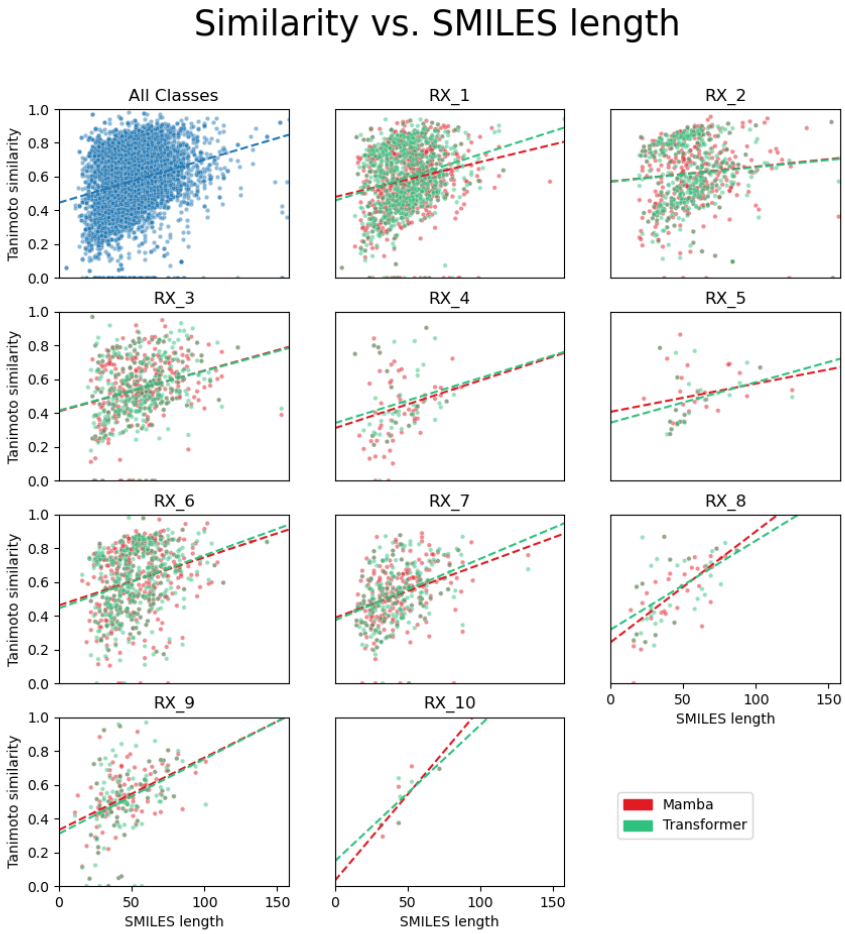


Figure 5.13: Tanimoto similarity vs. SMILES length for all incorrectly predicted molecules for the Mamba and Chemformer models.

	Mamba Mol Acc.	Transformer Mol Acc.	Absolute Count
One Reactant	0.419	0.475	1455
Two Reactants	0.481	0.519	3522
Three Reactants	0.2	0.266	15

Table 5.1: Molecular accuracy for the number of reactants that need to be predicted. The number of products is allways one.

Another important feature of the dataset is the number of predicted reactants. For most of the SMILES this is two. This means that two reactants react into one product. For some reactions this is not the case. The reaction could also use only one reactant and a solvent that is not in the SMILES string. Or, for example, three reactants that react into a single product. Figure 5.1 shows the molecular accuracy for these described groups. This shows that both Mamba and Transformer do best with the most common reaction of two reactants to one product. For the other two groups, both models score

significantly lower, with Transformer seeming to adapt better to the single reactant case than Mamba. This loss of performance could indicate that both models could be optimised for performance in the single reactant case.

5.4 Reaction Classes

As mentioned in Section 2.2, the USPTO-50k dataset consists of ten different reaction classes. Figure 5.14 shows the performance of both models for each reaction class. This clearly shows that not all reaction types are equally well learned. For example, for reaction class 10, both models achieve around 70%, whereas only 30% of reactions from reaction class 9 are predicted correctly. Figure 5.14 also shows that the Mamba and Transformer architectures perform quite similarly for most reaction classes, but for a few, e.g. classes 4 and 5, Transformer outperforms Mamba by more than 10%. This, like the SMILES length distribution, is not balanced, as can be seen in Figure A.1, and 5.13.

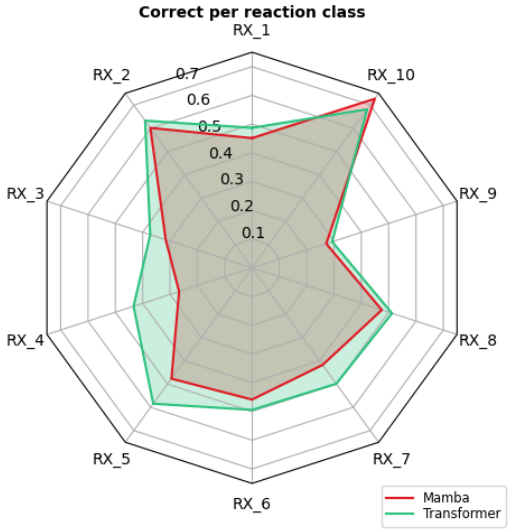


Figure 5.14: Diagram showing the performance of the Mamba and Chemformer models for each reaction class.

It is important to note that no information about the reaction class was given to the models during training. This would further increase accuracy, but is also not realistic in the application of reaction prediction. Without any reaction class information, the model had to learn which reaction type can be applied to which molecules.

For some reaction classes a hypothesis can be made about the performance of both models. For example, reaction class 3. This is the carbon-carbon bond formation class. It consists of organic reactions that form new covalent bonds between two carbon atoms. The poor performance of both models for this reaction class could be explained by the very broad definition of this class. Because a carbon-carbon bond can be a single or double bond, it can link many different parts of a molecule and is generally the most common bond in organic chemistry. This makes this reaction class very diverse and possibly not easy to predict. Unlike protecting groups, which only change or interfere with functional groups, C-C bond formation can change large parts of the molecule or, for example, split long molecules in half. Examples of this reaction class can be found in Figure A.8.

Reaction class 4 is also difficult to predict. This is the class of heterocycle formation. A heterocycle is a cyclic organic compound containing at least one element other than carbon. For example nitrogen, oxygen or sulfur. The formation of these heterocycles can be difficult to learn for reaction prediction models using the SMILES representation, as the cycles have to be split in SMILES. This is explained in more detail in Section 2.2. The splitting of cycles may explain why the Mamba and the Transformer struggle with this reaction. An example of random reactions from this class can be found in the appendix Figure A.9.

Reaction class 9 is the class of functional group interconversions. This includes all organic reactions in which one functional group is converted into another by substitution, addition, elimination, oxidation or reduction. Similar to carbon-carbon formation, this reaction can be of varying complexity because there are many different ways for a functional group to react. An example of a random reaction from this class can be found in the appendix Figure A.10.

Both models perform very well on reaction class 10. This is the functional group addition class. Unlike functional group interconversion, this reaction adds a functional group to a position on a molecule. This is much simpler than converting an existing functional group. An example of a random reaction in this class can be found in the appendix Figure A.11.

5.5 Model Agreement Analysis

Another way to compare the performance of the Mamba architecture with the Chemformer model is to analyse the predicted SMILES directly. Since both models were evaluated on exactly the same test set, it is possible to directly compare which SMILES were correctly predicted and which were not. Figure 5.15 shows this evaluation in the form of a confusion matrix. The groups where both models predicted the same are coloured light and dark green. This is not to be confused with the commonly coloured green true positive and true negative groups of a real confusion matrix where one model made a correct prediction. Dark green in this case means that both models failed to predict the correct SMILES. This figure shows that the intersection of equally predicted SMILES in the test set is large. The two models disagree on 20% of the SMILES, while 80% are predicted exactly the same. The red and orange patches are of particular interest, as this is where the two models disagree and therefore learn different generalisations from exactly the same data. For 12.38% of the data set, The Transformer predicted the correct molecule, while Mamba failed. The reverse is only true for 8%. It is also important to note that an ensemble model, which combines all the correct predictions of both models, could theoretically outperform all state-of-the-art models on this task, reaching 85.6%. In practice, this is difficult to achieve because the task of identifying which model made the correct prediction is as difficult as the prediction itself. Especially considering that the insight from Figure 5.12 and 5.13 is that both models are very likely to predict candidates even if they are wrong, because they have only a few wrong tokens. An attempt to build such an ensemble model can be found in Figure 5.6.

As most of the evaluations in this chapter have analysed the performance either per SMILES length or per reaction class, this is also done for the agreement between the two models. Figure 5.16 shows the percentage of each group of the confusion matrix from 5.15 for each reaction class. Figure 5.17 does the same for the SMILES length.

		Mamba		
		Correct	Wrong	SUM
Transformer	Correct	1906 38.18%	618 12.38%	2524 50.5%
	Wrong	400 8.01%	2068 41.43%	2468 49.4%
	SUM	2306 46.1%	2686 53.8%	4992

Figure 5.15: Confusion matrix-like table visualising the intersections of the prediction results from Mamba and Chemformer.

We can clearly see from these plots that there are some reaction classes where most of the SMILES are predicted correctly by both models. Reaction class 10, for example, showed high scores for both models in this evaluation so far. This is also true for this visualisation. This figure also shows that the amount of agreement or disagreement varies quite a lot between the reaction classes. However, so far no correlation can be found between any of the scores for the reaction classes and the properties of the molecules. This leads to the conclusion that the different scores for the classes can only be explained by their chemical structure and reaction rules and not by any other chemical or linguistic properties outside the classification between reaction rules.

As already concluded in this section, the same cannot be said for the evaluation of different SMILES lengths and the model performance for them. Figure 5.17 supports this, as one can see a clear relationship between Mama/Chemformer agreement and sequence length. While the Transformer correctly predicts SMILES of all lengths and only slightly underperforms for longer ones, this figure shows a clear correlation between sequence length and the valuable contribution of using the Mamba architecture for this task instead of just using a Transformer. In fact, Mamba is only able to correctly predict molecules where the Transformer failed, for short to medium length SMILES.

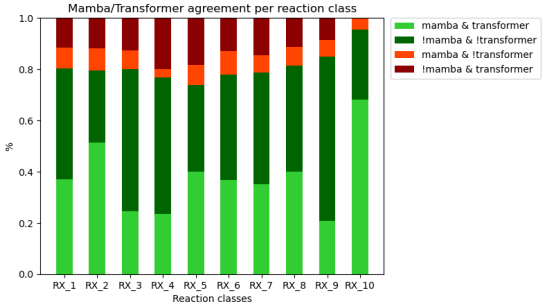


Figure 5.16: Stacked bar chart of the intersection percentages between the Mamba and Chemformer predictions for each reaction class.

Figure 5.17: Stacked line diagram of the intersection percentages between the Mamba and Chemformer predictions, sorted for the SMILES length.

5.6 Ensemble Model

As Figure 5.15 shows, the potential of a theoretical ensemble model is very high. The simplest ensemble model, which simply uses the prediction from the model with the highest confidence (added log probabilities for all tokens in the prediction), is not very good. While this technique performs better than the Mamba model alone, it is barely able to outperform the Chemformer model alone (Figure 5.2). Figure A.7 illustrates that a direct comparison of the probabilities for the generated predictions is not sufficient to determine which model is correct in its prediction.

The second ensemble design tested uses the average log probabilities during the decoding step to determine the next token. This design is explained in more detail in Figure 4.3. This is much more successful than the simple comparison of confidence scores. Figure 5.2 shows that although this model is still far from the theoretically possible 58%, it can demonstrate that an ensemble is more performant than either model individually.

Model	Molecular Acc.	Invalid Rate
Mamba	46.19%	0.9816%
Chemformer	50.54%	0.5596%
Ordinary Ensemble	50.66%	-
Mean-Logits Ensemble	51.56%	0.5596%

Table 5.2: Performance of the two base models and the two designs of ensemble models.

6 Conclusion

This work compared the Mamba and Transformer architectures on a very hard to learn and specific chemical task. The focus was set on the scalability of both models in terms of model size, inference time and performance. While the Transformer performed 2% better on almost all metrics, the Mamba model was 386% faster in inference and on par with the Transformer in training time. This shows that Mamba can be used in a wide variety of downstream tasks and that masked pre-training can be used equivalently to existing training routines. For the domain of chemical NLP tasks, the Mamba architecture may be a particularly good candidate, as inference time is a precious commodity in high-throughput analysis of the large chemical space of possible reactions and molecules. Further evaluations show that an ensemble model combining both architectures gives promising results that are better than either model alone. This is also supported by the results of Dao and Gu (2024), which showed that a mixture model with alternating attention and Mamba blocks outperforms the standard Transformer model. However, more research is needed on how to best combine the two models. Finally, this work shows that Mamba is also very compatible with other SOTA architectures that incorporate additional chemical information. This work has used the NAG2G architecture, whose encoder can be combined with the Mamba model. Although not trivial, this combination has produced reasonably good results, with great potential for further improvement.

Overall, the Mamba architecture is very versatile and can be used seamlessly with other transformer architectures. This work has also shown that it can be used for many non-trivial natural language processing tasks that require special features in the language models, such as selective copying or understanding of long distance relations.

6.1 Future Work

As this thesis shows, the implementation of both Mamba and transformer architectures can benefit from each other. Future work can improve on combining the two models to build an ensemble model. Possible options can be the use of joint beam decoding or the training of a new classifier model between the two models. Other approaches can

be the combined training of an ensemble model or a mixture model built from Mamba and attention blocks, which promise good results.

Furthermore the combination of Mamba and other advanced SOTA Models like NAG2G can be further improved. Promising ways of combining the encoder embeddings with the Mamba generation process are embedding wise concatenation of the encoder embeddings and the hidden states, or including the encoder embeddings directly after the atom embedding in the input to the Mamba model.

References

Dosovitskiy Alexey. 2020. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv: 2010.11929,* (cited on page 4).

Thomas Blaschke, Josep Arús-Pous, Hongming Chen, Christian Margreitter, Christian Tyrchan, Ola Engkvist, Kostas Papadopoulos, and Atanas Patronov. 2020. "REINVENT 2.0: an AI tool for de novo drug design." *Journal of chemical information and modeling* 60 (12): 5918–5922. (Cited on pages 2 sq.).

Regine S Bohacek, Colin McMartin, and Wayne C Guida. 1996. "The art and practice of structure-based drug design: a molecular modeling perspective." *Medicinal research reviews* 16 (1): 3–50. (Cited on page 1).

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. "Flashattention: Fast and memory-efficient exact attention with io-awareness." *Advances in Neural Information Processing Systems* 35:16344–16359. (Cited on page 18).

Tri Dao and Albert Gu. 2024. "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality." *arXiv preprint arXiv:2405.21060,* (cited on pages i, 24, 34, 44).

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. "Unified language model pre-training for natural language understanding and generation." *Advances in neural information processing systems* 32:13063–13075. (Cited on page 12).

Kurt LM Drew, Hakim Baiman, Prashanna Khwaounjoo, Bo Yu, and Jóhannes Reynisson. 2012. "Size estimation of chemical space: how big is it?" *Journal of Pharmacy and Pharmacology* 64 (4): 490–495. (Cited on page 1).

Peter Ertl. 2003. "Cheminformatics analysis of organic substituents: identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups." *Journal of chemical information and computer sciences* 43 (2): 374–380. (Cited on page 1).

Vendy Fialková, Jiaxi Zhao, Kostas Papadopoulos, Ola Engkvist, Esben Jannik Bjerrum, Thierry Kogej, and Atanas Patronov. 2021. "LibINVENT: reaction-based generative scaffold decoration for in silico library design." *Journal of Chemical Information and Modeling* 62 (9): 2046–2063. (Cited on page 3).

Aleix Gimeno, Maria Jose Ojeda-Montes, Sarah Tomas-Hernandez, Adria Cereto-Massague, Raul Beltran-Debon, Miquel Mulero, Gerard Pujadas, and Santiago Garcia-Vallve. 2019. "The light and dark sides of virtual screening: what is there to know?" *International journal of molecular sciences* 20 (6): 1375. (Cited on page 2).

Anirudh Goyal and Yoshua Bengio. 2022. "Inductive biases for deep learning of higher-level cognition." *Proceedings of the Royal Society A* 478 (2266): 20210068. (Cited on page 10).

Albert Gu and Tri Dao. 2023. "Mamba: Linear-time sequence modeling with selective state spaces." *arXiv preprint arXiv:2312.00752,* (cited on pages i, 4 sq., 13, 15, 17 sqq., 24).

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. "Hippo: Recurrent memory with optimal polynomial projections." *Advances in neural information processing systems* 33:1474–1487. (Cited on page 17).

Albert Gu, Karan Goel, and Christopher Ré. 2021. "Efficiently modeling long sequences with structured state spaces." *arXiv preprint arXiv:2111.00396,* (cited on pages 13 sqq., 17).

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. *Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers.* In *Advances in Neural Information Processing Systems,* 34:572–585. online. (Cited on pages 13, 17).

Jeff Guo, Franziska Knuth, Christian Margreitter, Jon Paul Janet, Kostas Papadopoulos, Ola Engkvist, and Atanas Patronov. 2023. "Link-INVENT: generative linker design with reinforcement learning." *Digital Discovery* 2 (2): 392–408. (Cited on page 3).

Mark Harris, Shubhabrata Sengupta, and John D Owens. *GPU Gems 3, Chapter 39: parallel prefix sum (SCAN) with Cuda.* (Cited on page 18).

Haris Hasic and Takashi Ishida. 2021. "Single-step retrosynthesis prediction based on the identification of potential disconnection sites using molecular substructure fingerprints." *Journal of Chemical Information and Modeling* 61 (2): 641–652. (Cited on page 6).

Alan Kai Hassen, Paula Torren-Peraire, Samuel Genheden, Jonas Verhoeven, Mike Preuss, and Igor Tetko. 2022. "Mind the Retrosynthesis Gap: bridging the divide between single-step and multi-step retrosynthesis prediction." *arXiv preprint arXiv:2212.11809,* (cited on pages 7, 20).

Haozhe Hu, Yongquan Jiang, Yan Yang, and Jim X Chen. 2023. *Enhanced template-free reaction prediction with molecular graphs and sequence-based data augmentation.* In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management,* 813–822. (Cited on pages 21 sq., 25).

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. 2012. "ZINC: a free tool to discover chemistry for biology." *Journal of chemical information and modeling* 52 (7): 1757–1768. (Cited on pages 24 sq.).

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. 2022. "Chemformer: a pre-trained transformer for computational chemistry." *Machine Learning: Science and Technology* 3 (1): 015022. (Cited on pages i, 5, 8, 12, 21, 24 sqq.).

Wengong Jin, Dr.Regina Barzilay, and Tommi Jaakkola. 2020. *Hierarchical Generation of Molecular Graphs using Structural Motifs.* In *Proceedings of the 37th International Conference on Machine Learning,* 4839–4848. (Cited on page 3).

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. *Junction tree variational autoencoder for molecular graph generation.* In *International conference on machine learning,* 2323–2332. (Cited on page 8).

Pavel Karpov, Guillaume Godin, and Igor V Tetko. 2019. *A transformer model for retrosynthesis.* In *International Conference on Artificial Neural Networks,* 817–830. Springer. (Cited on page 21).

Eunji Kim, Dongseon Lee, Youngchun Kwon, Min Sik Park, and Youn-Suk Choi. 2021. "Valid, plausible, and diverse retrosynthesis using tied two-way transformers with latent variables." *Journal of Chemical Information and Modeling* 61 (1): 123–133. (Cited on page 21).

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. 2020. "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation." *Machine Learning: Science and Technology* 1 (4): 045024. (Cited on page 8).

Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. *Neural speech synthesis with transformer network.* In *Proceedings of the AAAI conference on artificial intelligence,* 33:6706–6713. (Cited on page 4).

Min Htoo Lin, Zhengkai Tu, and Connor W Coley. 2022. "Improving the performance of models for one-step retrosynthesis through re-ranking." *Journal of cheminformatics* 14 (1): 15. (Cited on page 20).

Zaiyun Lin, Shiqiu Yin, Lei Shi, Wenbiao Zhou, and Yingsheng John Zhang. 2023. "G2GT: Retrosynthesis Prediction with Graph-to-Graph Attention Neural Network and Self-Training." *Journal of Chemical Information and Modeling* 63. (Cited on pages 22, 25).

Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. 1997. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings." *Advanced drug delivery reviews* 23 (1-3): 3–25. (Cited on page 1).

Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. 2017. "Retrosynthetic reaction prediction using neural sequence-to-sequence models." *American Chemical Society central science* 3 (10): 1103–1113. (Cited on page 20).

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. "Lost in the middle: How language models use long contexts." *Transactions of the Association for Computational Linguistics* 12:157–173. (Cited on page 35).

Hannes H Loeffler, Jiazhen He, Alessandro Tibo, Jon Paul Janet, Alexey Voronov, Lewis H Mervin, and Ola Engkvist. 2024. "Reinvent 4: Modern AI–driven generative molecule design." *Journal of Cheminformatics* 16 (1): 20. (Cited on page 3).

Asher Mullard et al. 2017. "The drug-makerś guide to the galaxy." *Nature* 549 (7673): 445–447. (Cited on page 1).

Kevin P Murphy. 2023. *Probabilistic machine learning: Advanced topics.* MIT press. (Cited on pages 9, 13 sq.).

Hitesh Patel, Wolf-Dietrich Ihlenfeldt, Philip N Judson, Yurii S Moroz, Yuri Pevzner, Megan L Peach, Victorien Delannée, Nadya I Tarasova, and Marc C Nicklaus. 2020. "SAVI, in silico generation of billions of easily synthesizable compounds through expert-system type rules." *Scientific data* 7 (1): 384. (Cited on page 6).

Michael Schlander, Karla Hernandez-Villafuerte, Chih-Yuan Cheng, Jorge Mestre-Ferrandiz, and Michael Baumann. 2021. "How much does it cost to research and develop a new drug? A systematic review and assessment." *Pharmacoeconomics* 39:1243–1269. (Cited on page 1).

Petra Schneider and Gisbert Schneider. 2016. "De novo design at the edge of chaos: Miniperspective." *Journal of medicinal chemistry* 59 (9): 4077–4086. (Cited on page 2).

Linde Schoenmaker, Olivier JM Béquignon, Willem Jespers, and Gerard JP van Westen. 2023. "UnCorrupt SMILES: a novel approach to de novo design." *Journal of Cheminformatics* 15 (1): 22. (Cited on page 8).

Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. 2018. ""Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models." *Chemical science* 9 (28): 6091–6098. (Cited on page 20).

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. 2019. "Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction." *American Chemical Society central science* 5 (9): 1572–1583. (Cited on page 21).

Marwin HS Segler, Mike Preuss, and Mark P Waller. 2018. "Planning chemical syntheses with deep neural networks and symbolic AI." *Nature* 555 (7698): 604–610. (Cited on page 20).

Seung-Woo Seo, You Young Song, June Yong Yang, Seohui Bae, Hankook Lee, Jinwoo Shin, Sung Ju Hwang, and Eunho Yang. 2021. *GTA: Graph truncated attention for retrosynthesis.* In *Proceedings of the AAAI Conference on Artificial Intelligence,* 35:531–539. 1. (Cited on page 22).

Vignesh Ram Somnath, Charlotte Bunne, Connor Coley, Andreas Krause, and Regina Barzilay. 2021. "Learning graph models for retrosynthesis prediction." *Advances in Neural Information Processing Systems* 34:9405–9415. (Cited on pages 20 sq., 25).

Dagmar Stumpfe and Jürgen Bajorath. 2020. "Current trends, overlooked issues, and unmet challenges in virtual screening." *Journal of chemical information and modeling* 60 (9): 4112–4115. (Cited on page 2).

Ruoxi Sun, Hanjun Dai, Li Li, Steven Kearnes, and Bo Dai. 2020. "Energy-based view of retrosynthesis." *arXiv preprint arXiv:2007.13437,* (cited on page 21).

Ruoxi Sun, Hanjun Dai, Li Li, Steven Kearnes, and Bo Dai. 2021. "Towards understanding retrosynthesis by energy-based models." *Advances in Neural Information Processing Systems* 34:10186–10194. (Cited on page 21).

Małgorzata Świst, Jarosław Wilamowski, Dariusz Zuba, Jolanta Kochana, and Andrzej Parczewski. 2005. "Determination of synthesis route of 1-(3, 4-methylenedioxyphenyl)-2-propanone (MDP-2-P) based on impurity profiles of MDMA." *Forensic science international* 149 (2-3): 181–192. (Cited on pages 6 sq.).

Igor V Tetko, Pavel Karpov, Ruud Van Deursen, and Guillaume Godin. 2020. "State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis." *Nature communications* 11 (1): 5575. (Cited on page 21).

Paula Torren-Peraire, Alan Kai Hassen, Samuel Genheden, Jonas Verhoeven, Djork-Arné Clevert, Mike Preuss, and Igor V Tetko. 2024. "Models matter: The impact of single-step retrosynthesis on synthesis planning." *Digital Discovery* 3 (3): 558–572. (Cited on page 20).

Zhengkai Tu and Connor W Coley. 2022. "Permutation invariant graph-to-sequence model for template-free retrosynthesis and reaction prediction." *Journal of chemical information and modeling* 62 (15): 3503–3513. (Cited on page 22).

Zhengkai Tu, Thijs Stuyver, and Connor W Coley. 2023. "Predictive chemistry: machine learning for reaction deployment, reaction development, and reaction discovery." *Chemical science* 14 (2): 226–244. (Cited on page 6).

Arnold Tustin. 1947. "A method of analysing the behaviour of linear systems in terms of time series." *Journal of the Institution of Electrical Engineers-Part IIA: Automatic Regulators and Servo Mechanisms* 94 (1): 130–142. (Cited on page 15).

A Vaswani. 2017. "Attention is all you need." *Advances in Neural Information Processing Systems,* 6000–6010. (Cited on pages 4, 10 sq.).

Venkat Venkatasubramanian and Vipul Mann. 2022. "Artificial intelligence in reaction prediction and chemical synthesis." *Current Opinion in Chemical Engineering* 36:100749. (Cited on page 3).

W Patrick Walters, Matthew T Stahl, and Mark A Murcko. 1998. "Virtual screening—an overview." *Drug discovery today* 3 (4): 160–178. (Cited on page 2).

Yue Wan, Benben Liao, Chang-Yu Hsieh, and Shengyu Zhang. 2022. "Retroformer: Pushing the limits of interpretable end-to-end retrosynthesis transformer." *arXiv preprint arXiv:2201.12475,* (cited on page 22).

Kai Wang, Bengbeng He, and Wei-Ping Zhu. 2021. *TSTNN: Two-stage transformer based neural network for speech enhancement in the time domain.* In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing,* 7098–7102. IEEE. (Cited on page 4).

David Weininger. 1988. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules." *Journal of chemical information and computer sciences* 28 (1): 31–36. (Cited on page 8).

Peter Willett. 2006. "Similarity-based virtual screening using 2D fingerprints." *Drug discovery today* 11 (23-24): 1046–1053. (Cited on page 38).

David S Wishart. 2007. "Introduction to cheminformatics." *Current protocols in bioinformatics* 18 (1): 1–14. (Cited on page 1).

Chaochao Yan, Qianggang Ding, Peilin Zhao, Shuangjia Zheng, Jinyu Yang, Yang Yu, and Junzhou Huang. 2020. "Retroxpert: Decompose retrosynthesis prediction like a chemist." *Advances in Neural Information Processing Systems* 33:11248–11258. (Cited on page 20).

Lin Yao, Wentao Guo, Zhen Wang, Shang Xiang, Wentan Liu, and Guolin Ke. 2024. "Node-aligned graph-to-graph: Elevating template-free deep learning approaches in single-step retrosynthesis." *Journal of the American Chemical Society Au* 4 (3): 992–1003. (Cited on pages 7, 22 sq., 25).

Kevin Zhang, Vipul Mann, and Venkat Venkatasubramanian. 2024. "G-MATT: Single-step retrosynthesis prediction using molecular grammar tree transformer." *American Institute of Chemical Engineers Journal* 70 (1): e18244. (Cited on page 21).

Shuangjia Zheng, Jiahua Rao, Zhongyue Zhang, Jun Xu, and Yuedong Yang. 2019. "Predicting retrosynthetic reactions using self-corrected transformer neural networks." *Journal of chemical information and modeling* 60 (1): 47–55. (Cited on pages 21, 52).

Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. 2022. "Root-aligned SMILES: a tight representation for chemical reaction prediction." *Chemical Science* 13 (31): 9023–9034. (Cited on page 22).

Appendices

A Appendices

A.1 USPTO-50K Dataset

Figure A.1 shows the distribution of reaction classes in the USPTO-50k dataset. As can be seen, it is very unbalanced. About 80% of all reactions in the dataset are of one of the four most common reaction types (rx1, rx2, rx3 and rx4).

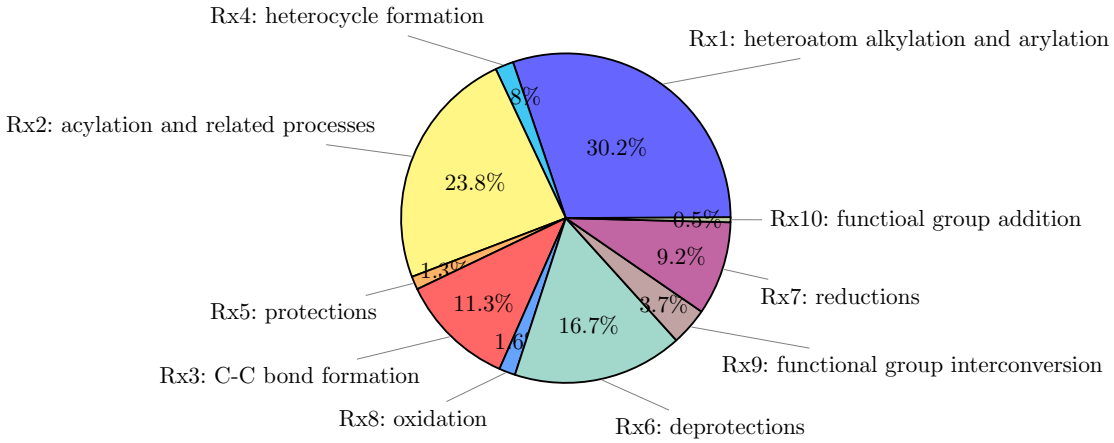


Figure A.1: Distribution of the reaction classes in USPTO-50k. [1]

Figure A.2 shows a example exerpt from the USPto-50k dataset in from of a csv file. Only the header and 3 lines of the file are shown. While the *id* and *class* columns are not used during training, the *reactants* > *reagents* > *product* column is used and split into its three parts. For the retrosynthetic prediction we use the product and reagents as input to the model and predict the reactants. Not every reaction uses reagents, so therefore they are often empty. This is also the case in Figure A.2.

1. Values taken from Zheng et al. (2019).

```
id,class,reactants>reagents>product
US07928231B2,UNK,"CC(C)(C)OC(=O)O[C:6]([O:5][C:2]([CH3
    :1])([CH3:3])[CH3:4])=[O:7].[CH3:8][C:9](=[O:10])[c
    :11]1[cH:12][cH:13][c:14]2[nH:15][cH:16][cH:17][c
    :18]2[cH:19]1>>[CH3:1][C:2]([CH3:3])([CH3:4])[O:5][
    C:6](=[O:7])[n:15]1[c:14]2[cH:13][cH:12][c:11]([C
    :9]([CH3:8])=[O:10])[cH:19][c:18]2[cH:17][cH:16]1"
US20090192322A1,UNK,"CC(C)(C)OC(=O)O[C:6]([O:5][C:2]([
    CH3:1])([CH3:3])[CH3:4])=[O:7].[CH3:8][c:9]1[cH
    :10][cH:11][c:12]([S:13](=[O:14])(=[O:15])[O:16][
    C@@H:17]2[CH2:18][NH:19][C@H:20]3[C@@H:21]2[O:22][
    CH2:23][C@@H:24]3[OH:25])[cH:26][cH:27]1>>[CH3:1][C
    :2]([CH3:3])([CH3:4])[O:5][C:6](=[O:7])[N:19]1[CH2
    :18][C@@H:17]([O:16][S:13]([c:12]2[cH:11][cH:10][c
    :9]([CH3:8])[cH:27][cH:26]2)(=[O:14])=[O:15])[C@@H
    :21]2[C@H:20]1[C@@H:24]([OH:25])[CH2:23][O:22]2"
US20080146614A1,UNK,"O=C1CCC(=O)N1[Br:1].[CH3:2][CH2
    :3][O:4][C:5](=[O:6])[c:7]1[n:8][n:9](-[c:10]2[cH
    :11][cH:12][c:13]([Cl:14])[cH:15][c:16]2[Cl:17])[c
    :18](-[c:19]2[cH:20][cH:21][c:22]([O:23][CH3:24])[
    cH:25][cH:26]2)[c:27]1[CH3:28]>>[Br:1][CH2:28][c
    :27]1[c:7]([C:5]([O:4][CH2:3][CH3:2])=[O:6])[n:8][n
    :9](-[c:10]2[cH:11][cH:12][c:13]([Cl:14])[cH:15][c
    :16]2[Cl:17])[c:18]1-[c:19]1[cH:20][cH:21][c:22]([O
    :23][CH3:24])[cH:25][cH:26]1"
```

Figure A.2: Example section from the USPTO-50K dataset.

A.2 Mamba and Transformer Comparison

All evaluations of the models in this thesis were performed without the knowledge of the reaction classes. Figures A.3 and A.4 were both taken after the pre-training and before the fine-tuning was done. Therefore, it is not possible to compare accuracy before fine-tuning (after pre-training) and after fine-tuning (after pre-training). Figure A.4 shows the performance on the masked prediction taks used for pre-training (for an explanation of the masked pre-training see 4.2). However, Figure 5.3, however shows the performance on the fine-tuning task, i.e. the single-step retrosynthetic reaction prediction. Therefore the two accuracy values between the figures are not comparable.

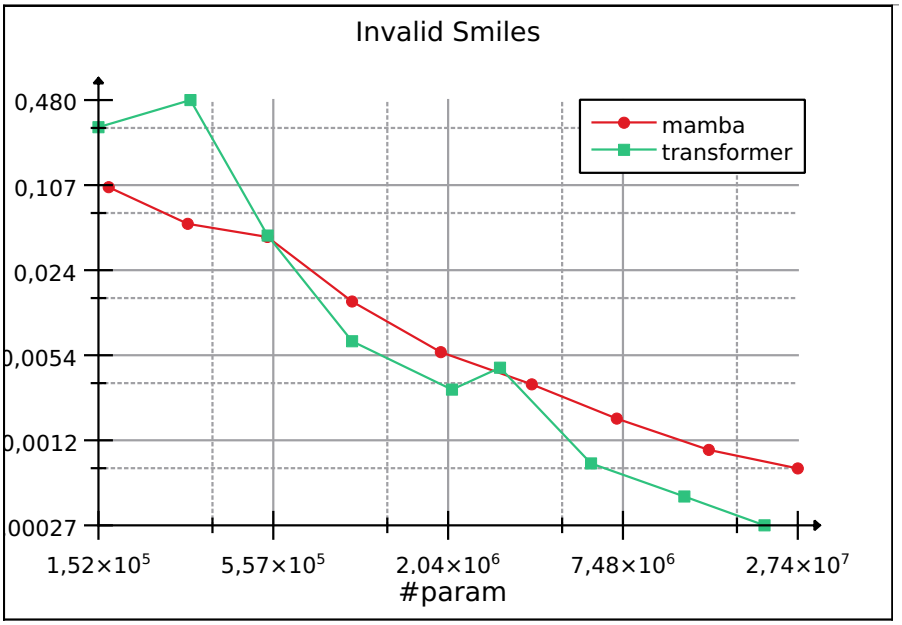


Figure A.3: Invalid rate of the Mamba and Chemformer model in relation to their model size. Measurements after pre-training without fine-tuning.

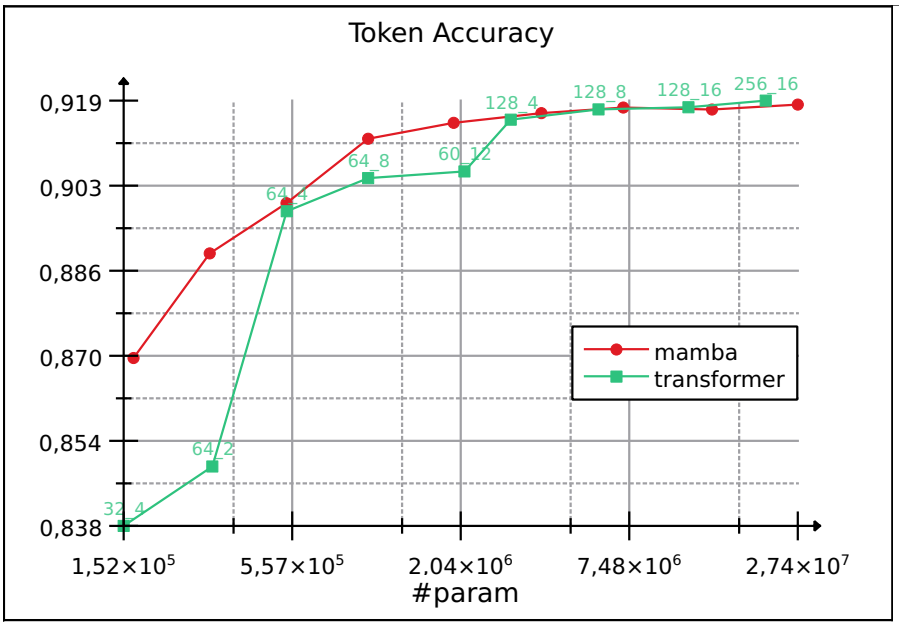


Figure A.4: Token accuracy in relation to the model size after pre-training and without fine-tuning.

Figure A.5 shows the average time taken by the model to predict the SMILES reactants for a reaction. This is not representative as the SMILES lengths of the reactions vary widely. Figure 5.7 shows the imbalance of the USPTO-50K data set with respect to SMILES length. Figures 5.5 and 5.6 show the inference time of the two models in a clear and unbiased way.

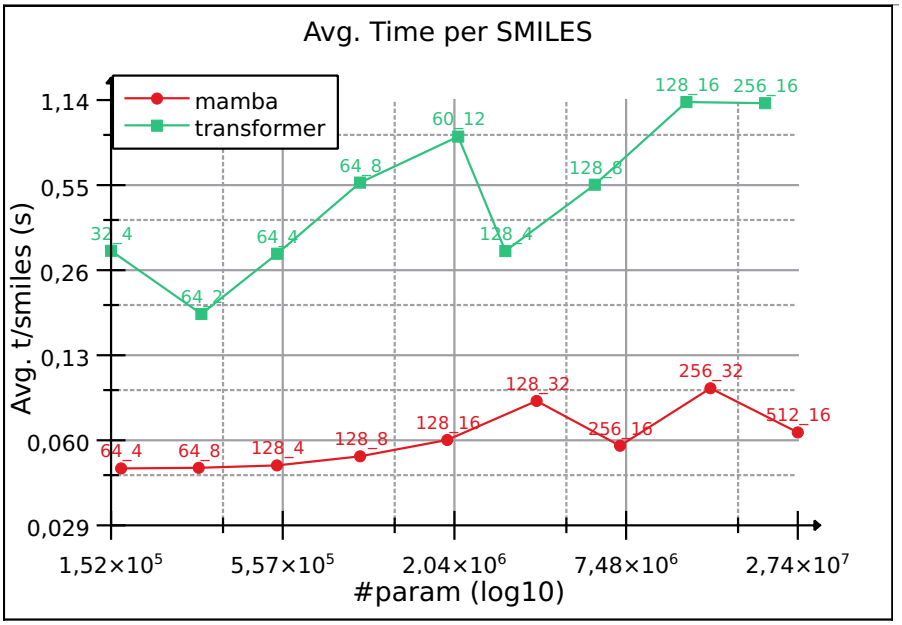


Figure A.5: Average inference time it takes each model to generate a SMILES in relation to their model size.

Figure A.6 shows the invalid rate compared to the SMILES length of the reaction. This information is binned in groups of 10. This shows that both models perform best for medium length reactions, while the invalid rate increases for shorter reactions and also for extremely long reactions.

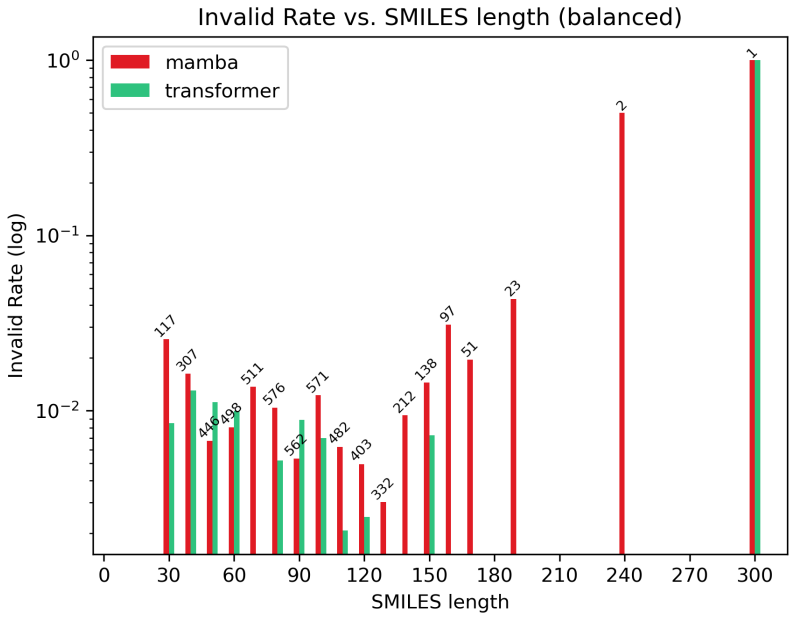


Figure A.6: Invalid rate in comparison to the SMILES length in the USPTO-50 test set.

Figure A.7 illustrates that a simple comparison of log probabilities between the transformer and Mamba models is not possible. If this were enough to decide which model's prediction was the more confident and therefore more likely to be correct. All the orange data points would be in the upper left triangle spanned by the diagonal through the origin. This would imply that for all predictions where Mamba predicted

correctly but Transformer did not, Mamba's prediction has a higher log probability than transformer's prediction. This is clearly not the case, so we can argue that a simple comparison of log probabilities is not possible.

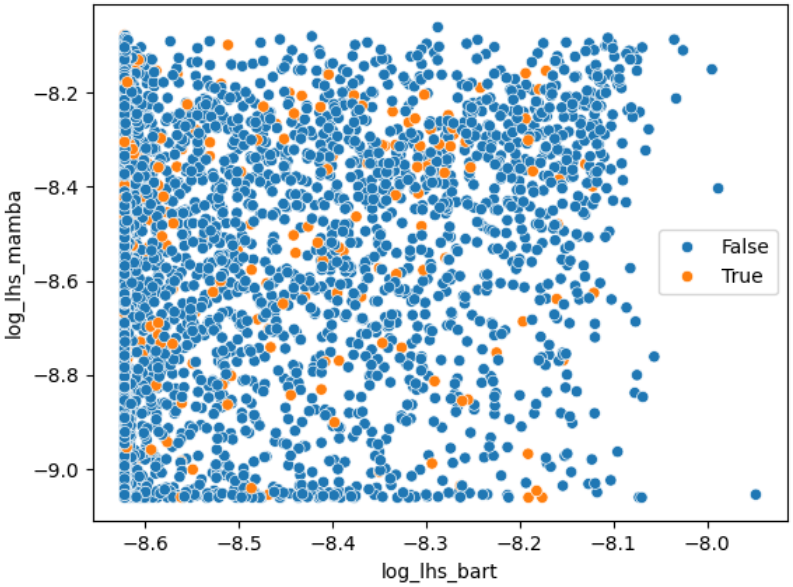


Figure A.7: Distribution of log probabilities for all SMILES in the test Dataset. The orange points are SMILES where Mamba predicted correct and Bart predicted wrong. Therefore, the log probability of these should be in the upper left triangle, if the simple comparison of log probabilities would be possible. Instead, there is no correlation found which could be learned by a simple classifier such as an SVM or a Random Forest.

A.3 Example Reactions

The following Figures A.8, A.9, A.10 and A.11 show example reactions from the different reaction classes. Each row in the figure is a separate reaction and its prediction from both models. The "Product" column is the input to the model, the "Target" is the ground truth to be predicted. The 'Mamba' and 'Transformer' columns are the predictions from each model.

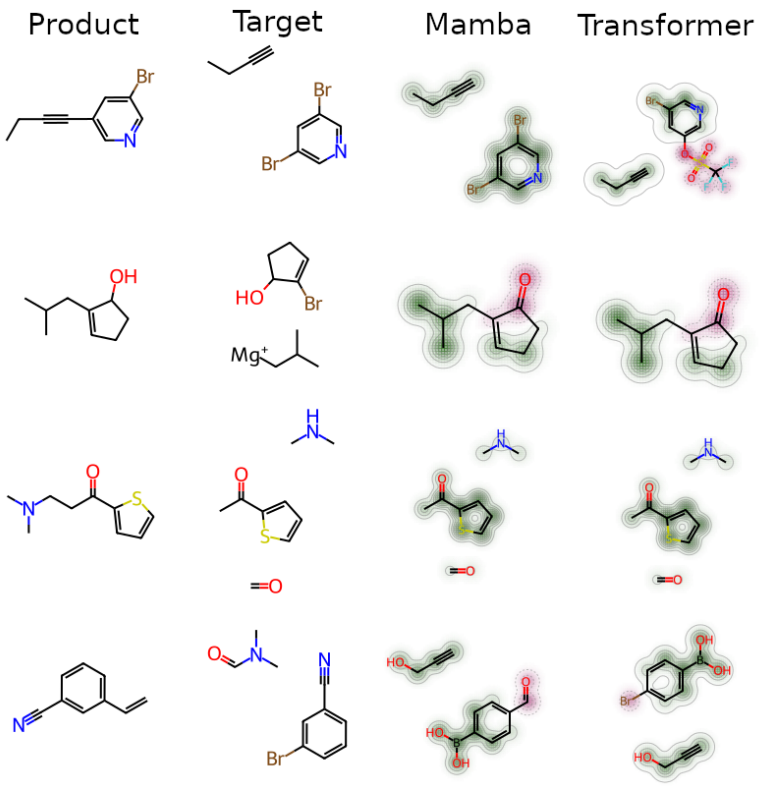


Figure A.8: Example reactants for the reactions class 3, Carbon-Carbon bond formations. For Mamba and Transformer reactant predictions each atom is colored in red/green for a negative/positive contribution to the similarity to the target.

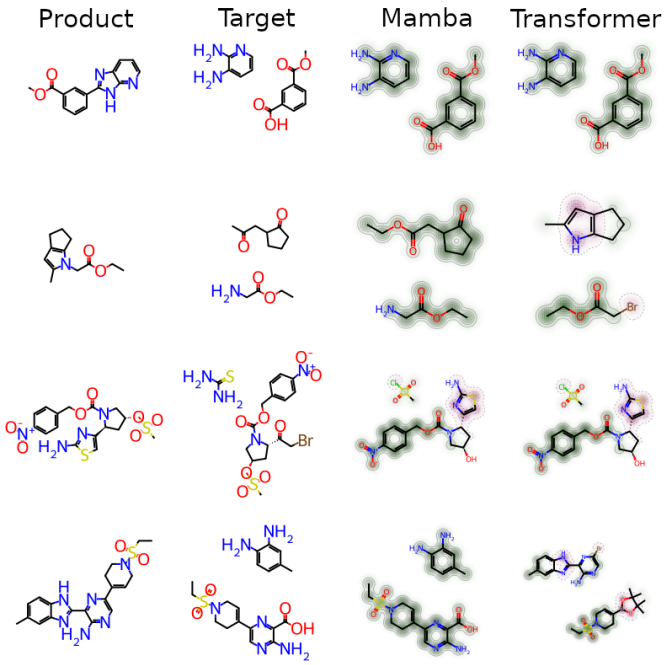


Figure A.9: Example reactants for the reactions class 4, heterocycle formation. For Mamba and Transformer reactant predictions each Atom is colored in red/green for a negative/positive contribution to the similarity to the target.

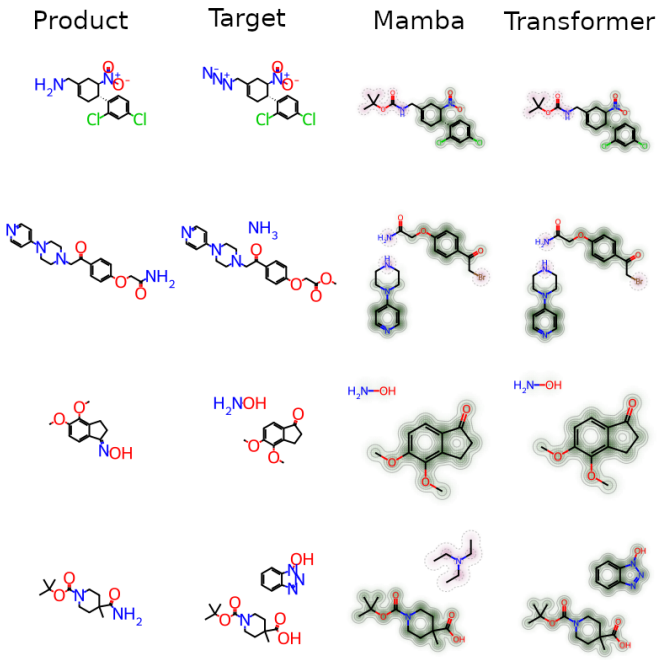


Figure A.10: Example reactants for the reactions class 9, functional group interconversions. For Mamba and Transformer reactant predictions each atom is colored in red/green for a negative/positive contribution to the similarity to the target.

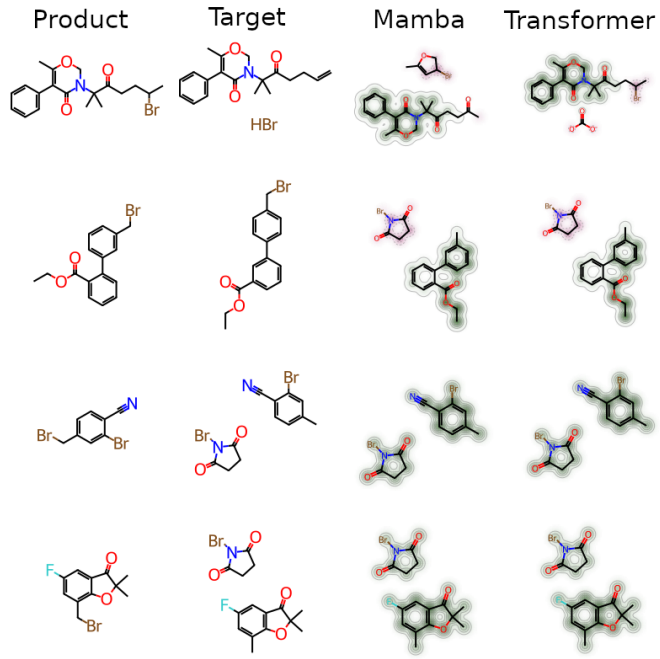
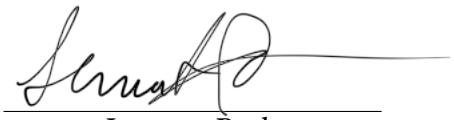


Figure A.11: Example reactants for the reactions class 10, functional group addition. For Mamba and Transformer reactant predictions each atom is colored in red/green for a negative/positive contribution to the similarity to the target.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.



Hamburg, den 24.09.2024 Lennart Roth

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 24.09.2024 Lennart Roth